'68' MICRO JOURNAL

$2.50 USA

Australia	A $ 4.00	New Zealand	NZ $ 4.00
Singapore	S $ 8.00	Hong Kong	H $20.00
	Malaysia	M $ 8.00	

VOLUME II ISSUE 10 • Devoted to the 68XX User • October 1980

"Small Computers Doing Big Things."

SYSTEMS – SOLUTIONS

If you have a problem that can be solved by a computer—we have a systems solution.

- Two central processors with maximum RAM capacities of 56K and 384 K bytes
- Three types of disk drives with capacities of 175K, 1.2M and 16M bytes
- Two dot matrix printers with 80 and 132 line capacity
- A Selectric typewriter interface and a daisy wheel printer

Match these to your exact need, add one or more of our intelligent terminals and put together a system from one source with guaranteed compatibility in both software and hardware.

Southwest Technical Products systems give you unmatched power, speed and versatility. They are packaged in custom designed woodgrain finished cabinets. Factory service and support on the entire system and local service is available in many cities.

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

UniFLEX™

Multi-User

UniFLEX is the first full capability multi-user operating system available for microprocessors. Designed for the 6809 and 68000, it offers its users a very friendly computing environment. After a user 'logs-in' with his user name and password, any of the system programs may be run at will. One user may run the text editor while another runs BASIC and still another runs the C compiler. Each user operates in his own system environment, unaware of other user activity. The total number of users is only restricted by the resources and efficiency of the hardware in use.

Multi-Tasking

UniFLEX is a true multi-tasking operating system. Not only may several users run different programs, but one user may run several programs at a time. For example, a compilation of one file could be initiated while simultaneously making changes to another file using the text editor. New tasks are generated in the system by the 'fork' operation. Tasks may be run in the background or 'locked' in main memory to assist critical response times. Inter-task communication is also supported through the 'pipe' mechanism.

Support

The design of UniFLEX, with its hierarchical file system and device independent I/O, allows the creation of a variety of complex support programs. There is currently a wide variety of software available and under development. Included in this list is a Text Processing System for word processing functions, BASIC interpreter and precompiler for general programming and educational use, native C and Pascal compilers for more advanced programming, sort/merge for business applications, and a variety of debug packages. The standard system includes a text editor, assembler, and about forty utility programs. UniFLEX for 6809 is sold with a single CPU license and one years maintenance for $450.00. Additional yearly maintenance is available for $100.00. OEM licenses are also available.

FLEX™

UniFLEX is offered for the advanced microprocessor systems. FLEX, the industry standard for 6800 and 6809 systems, is offered for smaller, single user systems. A full line of FLEX support software and OEM licenses are also available.

technical systems consultants, inc.
Box 2570, West Lafayette, IN 47906
(317) 463-2502 Telex 276143

'68' MICRO JOURNAL

Portions of text prepared using the following.

SWTPC 6800-6809-DMAF2-CDS1-CT82-Sprint 3
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216

EDITOR - WORD PROCESSOR
Technical Systems Consultants, Inc.
Box 2573, W. Lafayette, IN 47906
FLEX is TM of TSC

GIMIX Super Mainframe-Assorted memory boards
GIMIX Inc.
1337 West 37th Place
Chicago, Il 60609

Publisher: Don Williams Sr.

Executive Editor: Larry Williams

Subscriptions and Office manager
Mary Robertson

General Girl 'Friday'
Joyce Williams

Contributing Editors:

Dr. Jack Bryant
Dr. Chuck Adams
Dr. Theo Elbert
Dr. Jeffery Brownstein
Dale Puckett
Russell Gore
Ron Anderson
John Jordon

* CONTENTS *

FLEX USER NOTES.....10 Anderson

A BATTERY B/U BOARD.....12 Review

UCSD PASCAL™.....13 Puckett

BUSINESS PROGRAMMING.....17 Cagle

STYLOGRAPH W/P SYSTEM.....19 Pomerantz

POSITION INDEPENDENT CODE.21 Boyd

MORSRX (Morse code Rcv.)..24 Mayhugh W4MSR

HARDWARE HICCUP - MP-A2...30 Gass

DMAF-1/DMAF-2 ON 6800.....31 Gass

SOFTWARE MODS TO SWTBUG™..32 Hall

BIT BUCKET.....34 All of us

INTERFACING THE BITPAD....38 Taaffe

Send All Correspondence To:

'68' Micro Journal
3018 Hamill Rd.
PO Box 849
Hixson, Tennessee 37343

— Phone —
Office: 615-870-1993
Plant: 615-892-7544

'68' Micro Journal is published 12 times a year by '68' Micro Journal, 6131 Airways Blvd., Chattanooga, TN 37421. Second Class postage paid at Chattanooga, TN. Postmaster: Send Form 3579 to '68' Micro Journal, PO Box 849, Hixson, TN 37343.

1-Year $18.50 2-Year $32.50 3-Year $48.50

—ITEMS SUBMITTED FOR PUBLICATION—

(Letters to the Editor for Publication) All 'letters to the Editor' should be substantiated by facts. Opinions should be indicated as such. All letters must be signed. We are interested in receiving letters that will benefit or alert our readers. Praise as well as gripes is always good subject matter. Your name may be withheld upon request. If you have had a good experience with a 6800 vendor please put it in a letter. If the experience was bad put that in a letter also. Remember, if you tell us who they are then it is only fair that your name 'not' be withheld. This means that all letters published, of a critical nature, cannot have a name withheld. We will attempt to publish 'verbatim' letters that are composed using 'good taste.' We reserve the right to define (for '68' Micro) what constitutes 'good taste.'

(Articles and items submitted for publication) Please, always include your full name, address, and telephone number. Date and number all sheets. TYPE them if you can, poorly handwritten copy is sometimes the difference between go, no-go. All items should be on 8X11 inch, white paper. Most all art work will be reproduced photographically, this includes all listings, diagrams and other non-text material. All typewritten copy should be done with a NEW RIBBON. All hand drawn art should be black on white paper. Please no hand written code items over 50 bytes. Neatly typed copy will be directly reproduced. Column width should be 3¼ inches.

(Advertising) Any Classified: Maximum 20 words. All single letters and/or numbers will be considered one (1) word. No Commercial or Business Type Classified advertising. Classified ads will be published in our standard format. Classified ads $7.50 one time run, paid in advance.

Commercial and/or Business advertisers please write or phone for current rate sheet and publication lag time.

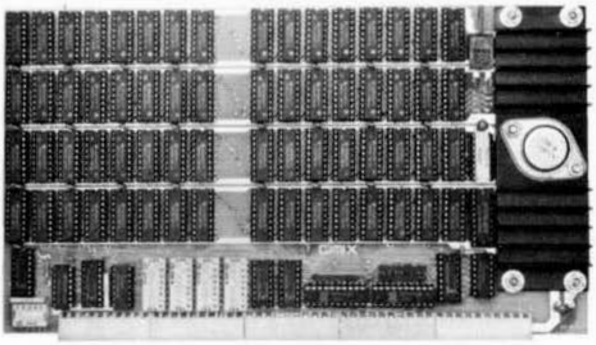
LOOK

WHAT'S COOKING on the FIFTY BUS

32K STATIC RAM BOARDS

Designed for use with:

★ Existing SS50 Systems ★ SS50C Extended Address Systems



- Assembled
- Burned In
- Tested

16K . . . $328.12
24K . . . $438.14
32K . . . $548.15

16K and 24K Versions are socketed for 32K and require only additional 2114's for expansion.

FEATURES:

- Decoding for 4 Extended Address Lines (allows memory decoding up to 1 megabyte)
- DIP-switch to set extended addressing or disable it
- 4 separate 8K blocks, addressable to any 8K boundary by DIP-switch
- Each 8K block may be individually disabled
- Write protect either of two 16K sections
- Low power consumption — uses 2114L low power RAMS
- Fully Socketed
- Gold Bus Connectors
- Guaranteed 2MHz operation

AND NOW . . . GIMIX OFFERS YOU A Choice of 6800 or 6809 CPU CARDS

You can order your system to fit your needs or select one of the below featured systems. Please contact the factory for further information and availability.

Add as much memory as you need using **GIMIX Static RAM Cards** for the utmost in reliability.

32K 6800 SYSTEM $1,694.59
Includes: Chassis, 6800 CPU, 32K RAM BOARD, I/O card

32K 6809 SYSTEM $1,844.69
Includes: Chassis, 6809 CPU, 32K RAM BOARD, I/O card

32K 6809 PLUS SYSTEM $1,994.79
Includes: Chassis, 32K RAM BOARD, I/O Card, and features our 6809 PLUS CPU Card with the Time of Day Clock option with battery back-up installed, as well as the 6840 Timer Package that provides 3 independent 16 bit counters.

This system also allows the following options to be added at additional cost:

- Battery back-up of the 1K RAM by substituting CMOS parts.
- A 9511 or 9512 Arithmetic Processor.
- GIMIX or SWTP Dynamic Address Translators.

EXPORT NOTES:

For 50Hz 230V C.V. POWER SUPPLY . . . . . Add $30.00
80 x 24 VIDEO BOARDS — Specify Format (No Added Charge)

On Orders under $250.00 for a Single Board, or Chips, please Add $30.00 Handling and we will ship Air Mail Prepaid. On all other orders we will ship via Emery Air Freight Collect, and we will charge no handling. All orders must be prepaid in U.S. Funds. Please note that foreign checks have been taking about eight weeks for collection, so we would advise wiring money or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, Account #73-32033. Visa or Master Charge also accepted.

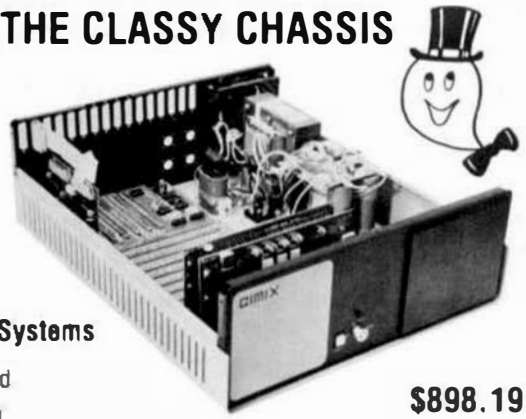
FACTORY PRIME STATIC RAMS

2114L 450 ns . . $5.90 300 ns . . $6.40 200 ns . . $6.90
4044 450 ns . . $5.90 250 ns . . $6.90

ADD $5.00 HANDLING ON ORDERS UNDER $200.00

GIMIX® and GHOST® are Registered Trademarks of GIMIX INC.

THE CLASSY CHASSIS



$898.19

- 25 amp (5V) ferro-resonant constant voltage power supply.
- Heavy weight aluminum cabinet with 3 position key switch, fan, and provisions for two 5" disk drives;
- 6800/6809 Mother Board, fifteen 50 pin and eight DIP-switch addressable 30 pin slots (gold plated pins), fully decoded.
- Baud rate generator on I/O section of Mother Board.

I/O BOARDS

for the 30 PIN BUS:

1 Port Serial (RS 232 or 20MA, current loop)	$ 88.41
2 Port RS 232 Serial	128.43
2 Port Parallel	88.42

for the 50 PIN BUS:

8 Port RS 232 Serial	288.40
8 Port RS 232 Serial with on board Baud Rate generator.	318.46
8 Port Parallel	198.45

BOTH 6809 SYSTEMS FEATURE OUR NEW TERMINAL BASED GMXBUG 09 SYSTEM MONITOR

GMXBUG 09 includes advanced debugging tools, utility, and memory manipulation routines.

Both 6809 Systems:

- ★ Can be reconfigured to allow use of other system monitors (DS-9 and S8UG-E)
- ★ Include 1K of Scratchpad RAM on the CPU
- ★ Allow optional software switching of system monitors.

2MHz 6809's at slight additional cost when they become available.

Phone, write, or see your dealer for details and prices on our broad range of Boards and Systems for the SS50/SS50C bus and our AC Power Control Products for all computers.

GIMIX INC.

The Company that delivers

Quality Electronic products since 1975.

1337 WEST 37th PLACE, CHICAGO, IL 60609
(312) 927-5510 • TWX 910-221-4055

SEE GHOST AD PAGES 44, 47, 52, 56

INTRODUCING

6809 SOFTWARE POWER TOOLS

BY MICROWARE®

OS-9™ LEVEL TWO MULTIUSER OPERATING SYSTEM

True multitasking, multiuser OS for timesharing or real-time control applications.

- Sophisticated memory management permits use of over one megabyte.
- Versatile, easy-to-use input/output supports multiple devices.
- UNIX™-like file structure including hierarchical directories, pipes, filters, and byte-addressable random access files.
- Provides log-on password protection and user file security.
- Can run on small, inexpensive systems with floppy disks and as little as 32K memory.

☐ $495.00*

OS-9™ LEVEL ONE OPERATING SYSTEM

Single-user, single-memory map compatible subset of Level Two for software development or stand-alone control applications.

- Versatile input/output system can support multiple devices using interrupt-driven, DMA, or program-controlled data transfer. Users can easily add additional I/O devices.
- Tape or disk-based versions available.
- Disk versions support UNIX™-like hierarchical directory structure and byte-addressable random-access files.
- Memory management for single address-space (up to 64K).

☐ Disk version $150.00*
☐ Tape version $95.00

BASIC09™ PROGRAMMING LANGUAGE SYSTEM

Extended BASIC language compiler/interpreter with integrated text editor and debug package. Runs standard BASIC programs or minimally-modified PASCAL programs.

- Permits multiple named program modules having local variables and identifiers. Modules are reentrant, position independent and ROMable.
- Additional control statements for structured programming: IF ... THEN ... ELSE, FOR ... NEXT, REPEAT ... UNTIL, WHILE ... DO, LOOP ... ENDLOOP, EXITIF ... ENDEXIT.
- Allows user-defined data types and complex data structures. Five built-in data types: byte, integer, 9 digit floating-point, string and boolean.
- Extremely fast program execution.
- Available on ROM, disk or cassette tape. Runs under OS-9™ Level One or Level Two.

☐ Disk or tape $195.00*

MICROSOFT 6809 BASIC

Standard Microsoft BASIC optimized for the 6809 and OS-9™.

- Four data types: integer, string, single precision and double precision floating point.
- Program trace and edit capabilities.
- Automatic line numbering and renumbering.
- Supports random and sequential file I/O. Full PRINT USING for formatted output.

☐ Disk or tape $250.00

OS-9™ TEXT EDITOR

Minimum-keystroke macro text editor useful for text preparation or interactive word processing.

- User-defined macros with parameters permit virtually unlimited command expansion. Macros can be saved, loaded and edited.
- Buffer, line and character oriented commands.
- Search, change and extend operations.
- Permits multiple input/output files.

☐ Disk or tape $75.00
☐ ROM set (2716) $90.00

OS-9™ INTERACTIVE ASSEMBLER

Compact Motorola compatible assembler for machine language program development.

- Operates in "batch" mode or interactive line-by-line mode.
- Facilities for generation of OS-9™ memory modules and system calls.
- Formatted listings include syntax *and* context error checking.
- Runs on OS-9™ Level One or Level Two.

☐ Disk or tape $75.00
☐ ROM set (2716) $90.00

OS-9™ INTERACTIVE DEBUGGER

Facilitates testing and debugging of machine- language programs.

- Includes common "monitor" functions: memory examine/change, breakpoints, display/change registers, hexadecimal arithmetic, etc.
- Access to system command interpreter.
- Available on ROM, disk or cassette tape.

☐ Disk or tape $35.00
☐ ROM (2716) $50.00

BASIC 09 is a trademark of Motorola. OS-9 is a trademark of Motorola and Microware®. UNIX is a trademark of Bell Telephone Laboratories.

Most software is available on ROM, diskette and tape in versions for many popular 6809 computers. Source listings and yearly maintenance/update service are sold separately for most programs.

*Specify manufacturer and type of CPU and I/O controllers. Contact Microware® for specific availability.

MICROWARE®

Microware Systems Corporation
5835 Grand Avenue, Box 4865
Des Moines, Iowa 50304
(515) 279-8844

INNOVATION AND PERFORMANCE

6800 SOFTWARE SUPER POWER

BY MICROWARE®

A/BASIC COMPILER

This BASIC compiler generates pure, fast, efficient 6800 machine language from easy to write BASIC source programs. Uses ultra-fast integer math, extended string functions, boolean operators and real-time operations. Output is ROMable and *runs without any run-time package.* Disk versions have disk I/O statements and require 12K memory and host DOS. Cassette version runs in 8K and requires RT/68 operating system.

☐ Disk Extended Version 2.1 SSB or FLEX* Diskette $150.00
☐ Cassette Version 1.0. $65.00

A/BASIC SOURCE GENERATOR

An "add-on" option for A/BASIC Compiler disk versions that adds an extra third pass which generates a full assembly-language output listing *and* assembly language source file. Uses original BASIC names and inserts BASIC source lines as comments.

☐ SSB or FLEX* Diskette $75.00

A/BASIC INTERPRETER

Here it is - a super-fast A/BASIC compiler! Now you can interactively edit, execute and debug A/BASIC programs with the ease of an interpreter - then compile to super efficient machine language. Also a superb stand-alone applications and control- oriented interpreter. Requires 8K RAM. The cassette version is perfect for Motorola D2 kits.

☐ Cassette, SSB or FLEX* Diskette $95.00

LISP INTERPRETER

The programming Language LISP offers exciting new possibilities for microcomputer applications. A highly interactive interpreter that uses list-type data structures which are simultaneously data and executable instructions. LISP features an unusual structured, recursive function-oriented syntax. Widely used for processing, artificial intelligence, education, simulation symbolic, and computer-aided design. 6800 LISP requires a minimum of 12K RAM.

☐ Cassette, SSB or FLEX* Diskette $75.00

*FLEX is a trademark of Technical Systems Consultants

RT/68 REAL TIME OPERATING SYSTEM

MIKBUG — compatible ROM that combines an improved monitor/debugger with a powerful multitasking real-time operating system. Supports up to 16 concurrent tasks at 8 priority levels plus real time clock and interrupt control. Thousands in use since 1976 handling all types of applications. Available on 6830 (MIKBUG-type) or 2708 (EPROM-type) ROM. Manual is a classic on 6800 real-time applications and contains a full source program listing.

☐ RT68MX (6830) $55.00
☐ RT68MXP (2708) $55.00

6800 CHESS

A challenging chess program for the 6800. Two selectable difficulty levels. Displays formatted chess boared on standard terminals. Requires 8K memory. Machine language with A/BASIC source listing.

☐ Cassette, SSB or FLEX* Diskette $50.00

Our software is available for most 6800 systems on cassette or diskette unless otherwise noted. Phone orders welcomed. We accept MASTERCHARGE and VISA. We try to ship orders within 24 hours of receipt. Please call or write if you require additional information or our free catalog. Microware® software is available for OEM and custom applications.

MICROWARE®
Microware Systems Corporation
P.O. Box 4865, Des Moines, IA 50304
(515) 279-8844

MICROSOFT DISK EXTENDED BASIC FOR OS-9™

Microsoft Basic is by far the world's most popular Basic language — and a vast library of applications software written for it can be used on your system: business packages, scientific, engineering and educational programs, games, etc. The OS-9™ version of Microsoft Basic is a full implementation of Release 5.0 with the following features:

- Four variable types: Integer, String, Single Precision Floating Point (7 Digit), Double Precision Floating Point (16 Digit).
- Full PRINT USING for formatted output (includes asterisk fill, floating $, scientific notation, trailing sign, comma insertion).
- Trace Facilities for program debugging.
- Extensive program editing facilities via EDIT command.
- Matrices with up to 255 dimensions.
- IF/THEN/ELSE and WHILE/WEND for structured programming.
- Automatic Line numbering and renumber.
- Dynamic string space allocation.
- Random and sequential file I/O with variable length records.
- Protected files can be saved in coded binary format.
- CHAIN and COMMON statements — programs may be linked together and share common variables.

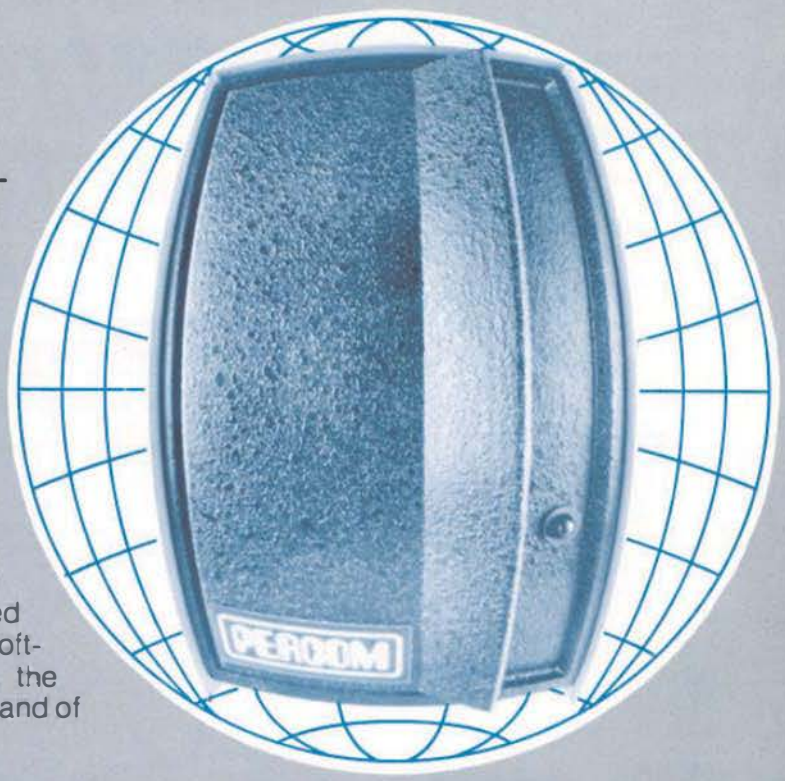
This version of Microsoft Basic is not just a reassembled 6800 Basic — it has been enhanced to take full advantage of the 6809 and OS-9™ superior capabilities. It is also a reliable Basic that you can count on for your important programs.

☐ Microsoft Extended Basic Release 5.0 for OS-9™ $250.00
☐ **Also available:** Standard Microsoft 6800 or 6809 Basic Release 4.51 for Flex®. Many features of OS-9™ version. $250.00 ™Trademark of Technical Systems Consultants

MICROWARE®
5835 Grand, Box 4865, Des Moines, IA 50304 • (515) 279-8844

In the world of 6800 Microcomputing there is only one Universal Mini-Disk System . . .

the PERCOM LFD-400™ with SOFTRAN™



Made possible by SOFTRAN™, an innovative $24.95 translator program, the reliable Percom LFD-400™ has just been upgraded to the first universal mini-disk storage system.

Suddenly the two worlds of 6800 minidiskette software become one. Because the LFD-400™ with SOFTRAN™ can read either soft-sectored or hard-sectored disks.

And owning an LFD-400/SOFTRAN system means you can run minidiskette programs from the enormous combined selection of all of the principal 6800 software houses — TSC, Computerware, the Software Works, Hemenway Associates and of course Percom.

Available in versions for mini FLEX†, FLEX 2.0† and Smoke Signal Broadcasting Company's DOS, SOFTRAN™ copies soft-sectored minidiskettes track-for-track onto hard-sectored minidiskettes. If the source disk includes a FLEX† or 'Smoke' DOS, SOFTRAN™ is used to modify the operating system to function with the Percom LFD-400™.

SOFTRAN™ is supplied on a minidiskette along with utilities for only $24.95. A users manual is included. You must indicate whether SOFTRAN™ is to be used for mini FLEX†, FLEX 2.0† or Smoke's DOS.

The Percom LFD-400™ mini-disk system sells for only $599.95, complete with: (1) the drive, drive electronics and Percom's rugged PS-401 power supply all in a finished enclosure, (2) a demonstrably superior controller PC card featuring an explicit data/ clock separation circuit, MPX, a remarkable 2K DOS, and provision for 1K extra PROM, (3) an interconnecting cable and (4) a 70-page users manual.

Also available: Upgrade kits for SWTP or 'Smoke' mini-disk drive systems. Kit includes LFD-400™ controller, MPX DOS & SOFTRAN™. Only $224.95.

Available soon!
SOFTRAN™ for Percom's 77-track LFD-800™ mini-disk system; SOFTRAN/9™ for 6809 FLEX† files and programs.

™ trademark of Percom Data Company, Inc.

† trademark of Technical Systems Consultants, Inc.

PERCOM

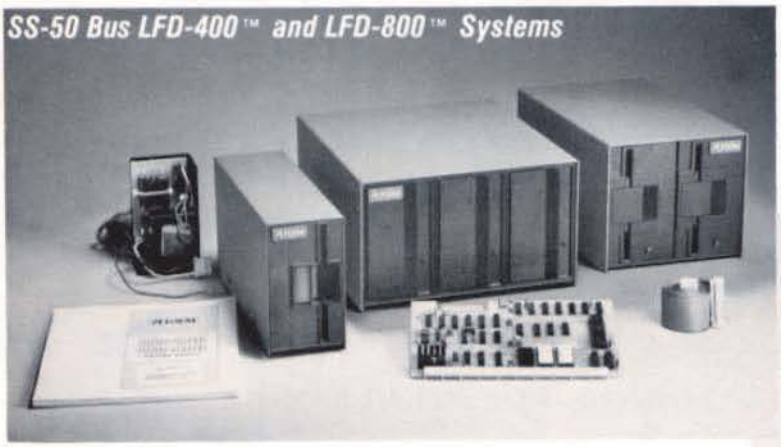
PERCOM DATA COMPANY, INC.
211 N. KIRBY GARLAND, TEXAS 75042
(214) 272-3421

Percom *'peripherals for personal computing'*

To place an order or request additional literature call toll-free 1-800-527-1592. For technical information call (214) 272-3421. Orders may be paid by check, money order, COD or charged to a VISA or Master Charge account. Texas residents must add 5% sales tax.

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE

A Few Extraordinary Products for Your 6800/6809 Computer

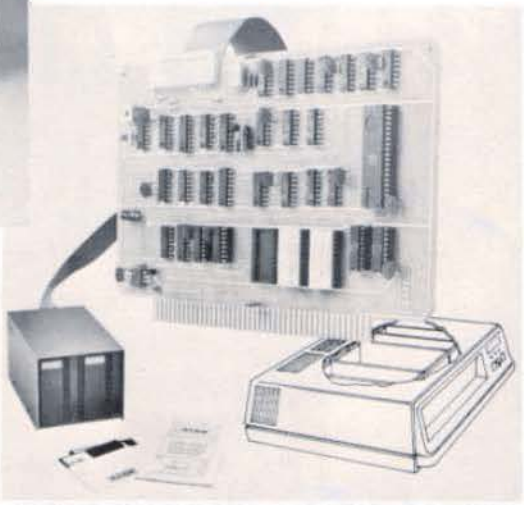


From Percom . . .

Low Cost Mini-Disk Storage in the Size You Want

Percom mini-disk systems start as low as $599.95, ready to plug in and run. You can't get better quality or a broader selection of disk software from any other microcomputer disk system manufacturer — at any price!

Features: 1-, 2- and 3-drive systems in 40- and 77-track versions store 102K- to 591K-bytes of random access data on-line • controllers include explicit clock/data separation circuit, motor inactivity time-out circuit, buffered control lines and other mature design concepts • ROM DOS included with SS-50 bus version — optional DOSs for EXORciser* bus • extra PROM sockets on-board • EXORciser* bus version has 1K-byte RAM • supported by extended disk operating systems; assemblers and other program development/debugging aids; BASIC, FORTRAN, Pascal and SPL/M languages; and, business application programs.



EXORciser Bus LFD-400EX™ -800EX™ Systems*

Versatile Mother Board, Full-Feature Prototyping Boards

Printed wiring is easily soldered tin-lead plating. Substrates are glass-epoxy. Prototyping cards provide for power regulators and distribute capacitor bypassing, accommodate 14-, 16-, 24- and 40-pin DIP sockets. Prototyping boards include bus connectors, other connectors and sockets are optional.

MOTHERBOARD — accommodates five SS-50 bus cards, and may itself be plugged into an SS-50 bus. Features wide-trace conductors. Price: $21.95

SS-50 BUS CARD — accommodates 34- and 50-pin ribbon connectors on top edge, 10-pin Molex connector on side edge. Price: $24.95.

SS-30 BUS CARD — 1¼-inch higher than SWTP I/O card, accommodates 34-pin ribbon connector and 12-pin Molex connector on top edge. Price: $14.95.

The SBC/9™, A "10" By Any Measure.

The Percom SBC/9™ is an SS-50 bus compatible, stand-alone Single-Board Computer. Configured for the 6809 microprocessor, the SBC/9™ also accommodates a 6802 without any modification. You can have state-of-the-art capability of the '09. Or put to work the enormous selection of 6800-coded programs that run on the '02.

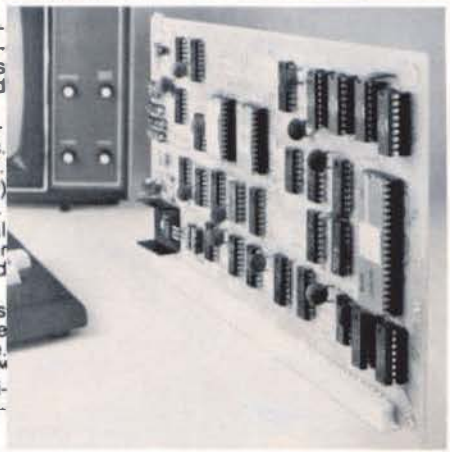
The SBC/9™ includes PSYMON™, an easily extended 1-Kbyte ROM OS. Other features include:

- Total compatibility with the SS-50 bus. Requires no changes to the motherboard, memory or I/O.
- Serial port includes bit-rate generator. RS-232-C compatible with optional subminiature 'D' connector installed. 10-pin Molex connector provided.
- Eight-bit, non-latched, bidirectional parallel port is multi-address extension of system bus. Spans a 30-address field; accommodates an exceptional variety of peripheral devices. Connector is optional.
- Include 1-K yte of static RAM.
- Costs only $199.95 with PSYMON™ and comprehensive users anual that includes source listing of PSYMON™.

The Electric Window™: Instant, Real-Time Video Display Control

Memory residency and outstanding software control of display format and characters make this SS-50 bus VDC card an exceptional value at only $249.95. Other features:

- Generates 128 characters including all ASCII displayable characters plus selected Greek letters and other special symbols.
- Well-formed, easy-to-read 7x12-dot characters. True baseline descenders.
- Character-store (display) memory included on card.
- Provision for optional character generator EPROM for user defined symbols.
- Comprehensive users manual includes source listing of Driver software. Driver — called WINDEX™ — is also available on mini-diskette through the Percom Users Group.



™ trademark of Percom Data Company, Inc.
* trademark of the Motorola Corporation.
Prices and specifications subject to change without notice.

PERCOM

PERCOM DATA COMPANY, INC.
211 N. KIRBY GARLAND, TEXAS 75042
(214) 272-3421

Products are available at Percom dealers nationwide. Call toll-free, 1-800-527-1592, for the address of your nearest dealer, or to order direct.

SOFTWARE...

MSI has MORE!

WE INVITE YOU TO LOOK AT OUR NEW SOFTWARE CATALOG WHICH OFFERS NEW PROGRAMS FOR YOUR 6800 SYSTEM.

•All FLEX™ Programs from TSC are now available for MSI Computer Systems.

•MULTI-DISK FLEX™ from MSI allows the use of any combination of MSI disk devices to be used simultaneously, including the HD-8/R 10 megabyte drive.

•SORT/MERGE Program can be used manually or within other BASIC or assembler programs to perform high speed sorts of data files.

•Hemenway Associates Software Products for use under FLEX™ are available on the MSI System.

•TRS-80/MICROSOFT BASIC - MSI BASIC Translator allows MSI users to run the large library of basic programs written for the TRS-80 and other similar systems.

•SOFTWARE LIBRARY Programs keep track of all diskette and hard disk directories, giving alphabetical listings of available programs.

•SDOS Operating System.

•MULTI-USER/MULTI-TASKING SDOS Operating System allows any user to perform edits, assemblies, compilations, or program executions independently and simultaneously.

•All MSI software is supported on four (4) disk systems: quad density minifloppy, single and double density 8" floppy, as well hard disk systems.

•Complete BUSINESS APPLICATION PACKAGES including sales order entry, accounts receivable, inventory management, purchase order entry, accounts payable, and general ledger are available on MSI hard disk systems.

•PLOTTING PACKAGE gives daisy-wheel printers the capacity to perform graphics operations.

•LETTERWRITER Word Processing Software allows the use of daisy-wheel printers to generate documents and to handle correspondence automatically.

FLEX™ is a registered trademark of Technical Systems Consultants, Inc.

Send for your catalog today.

Midwest Scientific Instruments

220 W. Cedar • Olathe, Kansas 66061 • 913-764-3273

TWX 910 749 6403 (MSI OLAT)

Telex 42525 (MSI A OLAT)

HEAR YE!

HEAR YE! HEAR YE!

SOFTWARE ANNOUNCEMENT

JCP — NOW INCLUDES LIBJCP

Job Control Program
By Peter Murray

The JOB CONTROL PROGRAM (JCP) reads a text file that contains the necessary input for a program and then supplies this input to the program in the same manner that an operator would have normally entered it from the keyboard. The file containing the input is referred to as the procedure file, and the program receiving the input is referred to as the calling program. A procedure file contains input for such calling programs as FLEX™, FLEX utility commands, and other development software.

LIBJCP is used as a FLEX™ command within a procedure to load and execute another procedure. At the conclusion of the called procedure, control will return to the calling procedure and execution will resume at the line following the LIBJCP command.

JCP also provides for parameter substitution within the procedure file, special commands to control JCP program flow, and a means for recovery from processing errors. These features allow for commonly used file routines to be written as a generalized procedure that JCP will execute, unattended, simply by entering a single FLEX command.

PRICE $89.95

See Review in July '80 '68' Micro.

INTRO PRICE UNTIL OCT. 1, 1980 $39.95

AVAILABLE NOW! REMOTE

Intelligent Terminal Program
By Tom Speer

REMOTE allows use of your 6800 or 6809 system as a terminal to a remote computer. REMOTE gives you access to the new, inexpensive time-sharing systems designed for home computer users, such as MICRONET or THE SOURCE, or any other time-sharing system. You can also use REMOTE to talk to another microcomputer system. All you need is your FLEX based 6800 or 6809 disk system, a serial interface, a modem, and REMOTE.

REMOTE is flexible. You can control its many features and options thru local commands at any time. Output from the host system (from the phone line) may be directed to the console CRT, the system printer, or a disk file, or any combination of the three. You can change parity checking, control character definitions, etc. You can read files from disk to the phone line under user control. You also can issue any FLEX command while you are not receiving data.

REMOTE is a terminal program designed to give you the utmost in flexibility and convenience with as many different timesharing systems as possible. It was written by Tom Speer, P.E., an Electrical Engineer with over twelve years of data communications hardware and software experience.

: INTRO PRICE $39.95

NEW READTEST

English Text Analysis Program
By Dale Puckett

READTEST is a must for all writers and writing instructors. Reads prose from disk file and tells how well it was written. Reports number of lines, words, sentences, personal words, affixes, average sentence length. Individual reports pinpoint trouble areas. Overall index tells who can read it and who would print it. Fast 6800 object code. Runs in FLEX™. READTEST, a 6800 assembly language program written by 68 Micro Journal Contributing Editor Dale L. Puckett, is such an indispensable applications program for all writers, fiction or non-fiction, student or professional. READTEST reads your letters, short stories, news stories, reviews, novels, etc., from the disk file you prepared them on and tells you how well they were written. READTEST is a powerful tool designed to aid both the student writer and keep a running check on the experienced writer. It is a serious application of your 6800 microcomputer that will pay for itself with the first check you receive for a readable manuscript.

APPROX. 12K : INTRO PRICE $39.95

AVAILABLE SOON HELP

Help for FLEX™
By Frank Hogg

Forgot the command syntax? Type HELP! HELP keeps information at your fingertips. It eliminates the need to fumble through manuals and also helps beginners operate your computer.

The HELP interpreter with its 6+ commands reads information from one of three text files, so adding, modifying or deleting information is easy. You can also create your own HELPfile. Everybody needs a little HELP!

SPECIAL PRICE $24.95

ESTHER

An exercise in artificial intelligence
By Dale Puckett

ESTHER is Eliza+. Artificial intelligence in pure 6800 code. Source shows how. Object amazes friends. ESTHER: remembers names, drops them, uses players name, answers third person replies, echos keywords. 75+ keywords. 48+ sets of replies. Auto line length. Runs in FLEX™. Obeys TTYSET. ESTHER is both educational and fun !!!

APPROX. 16K : INTRO PRICE $39.95

FLEX is a trademark of Technical Systems Consultants, Inc.

master charge — VISA

FRANK HOGG
DENTAL LABORATORY
130 Midtown Plaza
700 East Water St.
Syracuse, N.Y. 13210
(315) 474-7856

All software is currently available on FLEX™ 2.0 5" disks or MSI FLEX™ 1.0 8" hard sectored disks. The package includes a users manual, the disk with object code AND FULLY COMMENTED SOURCE LISTING, a programming manual with information about the program and hints for changes, and where applicable example programs.
VISA and MC accepted. SOURCE TCF339.

ATTENTION PROGRAMMERS!
We are looking for quality software to market. Contact Frank Hogg.
*6809 software should be ready by the time this ad is out.
Call for availability.

* 8" soft sectored (DMAF type) should also be available. Please call.

FLEX* USER NOTES

BY: RONALD W. ANDERSON
3540 STRUBRIDGE COURT
ANN ARBOR, MI 48105

A DUMB ERROR

Here's a story about yours truly and his stupidity. As I have mentioned, I bought a SWTPC 6809 processor board for myself. I recently bought a second for the company, where I have a system nearly identical to my ho system. Recently we ordered TSC DEBUG for the '09, and it simply wouldn't run. Between the company and my personal system, we have duplicates of everything, and it wouldn't run on my system either. I found one combination of 6809 processor, board, and system in which debug would run properly, and in addition I found that other combinations would run for a few minutes. Armed with the fact that I had a "heat problem" and a can of circuit cooler, I proceeded to find that if I would cool a particular area of the processor board, debug would again run for a few minutes. I noted that it seemed to die a slow death, the output in the STEP mode slowing down and finally stopping.

After a half dozen false starts and positive identifications of the problem, I got a good dual trace scope from a friend and started to poke around. I have a friend, Paul Patrick who has just put together a system and has the '09 running. DEBUG ran fine on his system. To make a very long week of 2 AM's short, since I bought an assembled board, I didn't read the fine print on the Addendum sheet, and see the note that use of the processor board with an MP-8 or MP-82 board, it is necessary to add a pull-up resistor to the FIRQ line, which is the old UD2 line on the SS-50 bus. I noted action on the bus status output of the processor, and then that the FIRQ line was low (it is active low). One pull-up resistor, and the problem went away permanently on both systems. Apparently, DEBUG clears the interrupt mask and the interrupt was seen, causing the program to go west. While the processor was cold, the open interrupt input was in the OFF state, and the system worked until it warmed up. One combination of processor, b rd, and system worked though perhaps marginally with that input open. Possibly the mother board had some leakage to the supply voltage on that line and held the input high.

Before this discovery, I was ready to start writing nasty letters to SWTPC about their marginal design, and to go order someone else's '09 board for my system. I restrained myself from doing so until the problem was found and solved. As someone once said, if all else fails, read the instructions. I had noticed that once in a while, a program would go west. I assume that this will not happen with the problem solved. SWTPC, I take back all the nasty things I was thinking about you. I have never had any trouble that has been traced to a marginal design of anything from SWTPC. At our company, we also have an 8080 system that is now being used in a word processing application, running "Wordstar". The hardware in that system seems to fail once a month. I have brought about a few failures of my system by a slip of the probe, etc., but the "random" failures, ie. those that "just happened", have been limited to a few initially bad memory chips (marginal), the main rectifier bridge (one diode shorted) and a couple of problems with the SA-400 drives. One drive motor quit, and one plastic photo transistor holder broke (fixed with super glue in 5 minutes). I consider this to be very high reliability for nearly three years of heavy use.

Z-80 NOT SO FAST AFTER ALL

I may be forced to do a significant programming project in Z-80 (ugh) Assembler, due to a customer's requirements. That has led me to look at the Z-80 instruction set in a more serious manner than I had previously done. One of the "great" features of the Z-80 is the highly publicized Block Move instruction. Actually there are four such instructions, but let's just talk about the LDIR instruction. This is presented using Zilog Mnemonics. Apparently one has the choice of using the "old" 8080 mnemonics for all the 8080 instructions, and the Zilog for the Z-80 only instructions, or going all the way and using all Zilog mnemonics. As I understand the LDIR instruction, the HL register is loaded with the "FROM" address, the DE register with the "TO" address, and the BC register with the byte count for the move. Assuming that the values FROM, TO, and NBYTES have been assigned a value previously, the code for a block move would look like this.

```
LD HL,FROM
LD DE,TO
LD BC, NBYTES
LDIR
```

These four instructions set up the move, and the LDIR instruction moves a byte, decrements B and repeats until B is zero. The LD HL extended instruction takes 10 machine states or clock cycles. LD DE and LD BC, the same, and the LDIR 16 cycles except on the last loop pass (where it is exited), which takes 21 cycles. That totals 133 clock cycles for a 6 byte move, and the code is 11 bytes. Since I don't have a system running, I don't guarantee absolute accuracy in this analysis, but it is close.

Now, let's look at the 6809 equivalent. The setting up of the registers is the same, and if we use a subroutine for the move, it only has to appear once in the program, and will not significantly increase the size of the program.

```
LDU #FROM
LDX #TO
LOY #NBYTES
LBSR MOVE

MOVE LOA ,U+
 STA ,X+
 LEAY -1,Y
 BNE MOVE
 RTS
```

The program contains the same four line call as in the Z-80. Since LDY requires four bytes, and we use a LBSR which is a byte more than the LDIR instruction, it costs us 13 bytes rather than the 11 bytes in Z-80. The sub outline is 9 bytes long. According to TSC's Debug States Counter, this program moves 6 bytes in a total of 144 clock cycles. Our 6809's running at 2 MHZ execute this code as fast as a Z-80 running at 2 MHZ. We have some flexibility with the 6809 that is lost in the Z-80 implementation. If the block to be moved is shorter than 256 bytes we may use the B accumulator as a counter and reduce the code a little. If this is done, the number of clock cycles used is reduced to 122. Suppose MOVE were a subroutine in a Math package, in which all variables were 5 bytes. We could include an instruction LDB #5 in the subroutine, reducing the calling code further. We could also use a BSR rather than LBSR for any call within the range of the BSR, thus further reducing byte count and increasing speed. If MOVE were used a large number of ti s in a program, one could set up one of the SWI vectors to go to it and use a single byte call SWI3 from anywhere in the program. Furthermore, the 6809 code may easily be position independent.

On the basis of this comparison, we 6809 users can only say that we can do as well as the Z-80 given the same clock speed. A look at the Z-80 instruction set and clock cycles required does reveal one area where we are much better off, though. The Z-80 has an X and a Y Index Register. Most of the indexed instructions take 19 clock cycles. Some take 23! Our 6809 does an indexed instruction in varying numbers of clock cycles depending on the option. The shortest is a JMP indexed which takes 3 cycles. The longest that I can figure out is a LDYIn,R1 as in LDY[$1234,X] which takes 11 cycles. Something like a LDA ,X+ only takes 6 cycles. The Z-80 doesn't even offer the indirect or the post incre nt instructions. Smile.

6809 NOTES

A few months ago, I mentioned my 6809 floating point math package, which was in the debug phase. I have most of it working now, and it has passed all priliminary tests using the four functions. I have not had a chance to do thorough checking of the routines that convert the input ASCII or packed BCD numbers to floating point binary notation and the inverse, nor am I satisfied that all the functions are fully operational at this point. I need to get

the package into an application and have some numbers thrown at it for a while to give it a good test before I will publish it as fully operational. I was able to use the 8 by 8 bit multiply instruction for a triple precision multiply that uses only about 400 clock cycles for the multiplication and the moving of data onto the stack and the result back off of the stack. This does not include the above mentioned conversions of format. It does represent a time for a multiplication in a series of calculations on variables already in floating point format. This of course makes the Divide routine the "turtle" instruction. Of course you may speed up a calculation by using multiplies rather than divides wherever possible. For example, rather than dividing by PI, you could take the reciprocal of PI once in the program and then multiply by this value (about 0.318). If I get a few letters from readers expressi g interest in this package, we will publish it over a period of time.

LUCIDATA INTERVIEWED

I recently wrote Lucidata asking for some information about the principals of the company, having had some extended correspondence with them concerning their Pascal compiler. I received a biographical sketch of them from which I have prepared this "interview". The company consists of three people. Nigel and Eileen Bennee, and David Gibby. David wrote the Pascal compiler, and Nigel the P-code interpreter or Runtime package. The Bennee's are British and Dave Gibby Welsh. They have been in Holland for about 10 years, having gone there to work in "the Computer Division of an International Research Establishment." Nigel has a degree in Nuclear Physics, for which there is not much demand in England, so he went into computing. The background includes knowledge of electronics, so that Nigel understands both hardware and software. oth men bought "minimal 6800's" in 1978, and since they were both accustomed to working on large computers, didn't like working with BASIC at home. With the expenditure of much midnight oil, they developed version I on their small systems. Eileen indicates that she too has an education in Physics, and worked in computing at the research center until their family required her to stay home. Eileen started Lucidata with the idea of doing programming work on a consulting basis, which she indicates has been her main occupation. Someone suggested that they should market Pascal, and "that is when we started working 20 hours a day!" Eileen goes on to indicate that the response to their first version was very favorable, and that was the reason that they went on and developed the second. They have moved recently. Their new address is:

LUCIDATA
Duinsroosweg 10
2597 KJ The Hague
Netherlands

I inquired as to whether they would have dealers here, and they indicated that they want to maintain control over the production of the software. They indicate that they feel that a lot of software for the Micro market is badly supported and that they are trying to do better.

I have found further nice features of the Pascal, in my attempt to apply it to programs for balancing machines. One in particular is worth writing about here. The Pascal as supplied has the capability of supporting 8 output devices including the terminal as two (one for input, and one for output). This means that you can overlay the device tables with the addresses of your I/O routines for some special device, and then READ and WRITE to it just as *though* it were your printer or terminal. I used this device table to jump to an input and outpuf routine to do something that is at first glance downright "silly". I wrote a routine to write to and read from memory. There is in addition a procedure that allows you to poke a location that defines the area of memory to which you want to write. Why on earth would anyone want to do that, as I always say! Simple, my system will have some battery backed up RAM (CMOS), and I want to save several sets of machine set-up parameters for various parts. Pascal, being stack oriented does not have very well defined locations for variables, so this device lets me write to and read from fixed memory locations. The data is kept alive by the batteries, and next time the program is run, the machine operator c n retrieve his machine set-up by an ID number (one of 32), and run a part that he ran last week without going through a set-up procedure again. I could easily add a routine to allow writing to my Modem under program control while running pascal. If you haven't guessed, I am very pleased with the Pascal and the Company. (I have no financial interest or other connection with Lucidata.)

POLYNOMIAL REGRESSION

No, it's not a disease. If you have the book, Some Common Basic Programs by Borchers and Poole, you have probably paged past their program of that name. It sounds deep and complicated, but it is really not. I have used it in the past to find fast approximations for Scientific Functions, but recently managed to get it teamed up with TSC Extended BASIC. When you have 14 digit arithmetic available, this program can perform magic. What does it do? It generates an equation that produces an approximation to a function that you input to it. Suppose you have a group of data points that you want to describe with a mathematical function. This program, after you input the data points (X and Y coordinates of the points), will produce an equation or formula of the form A3X©3+A2X©2+A1X+A0. The A's are called the coefficients of the terms of the equation which is called a polynomial. It contains descending "powers" of X. The power of the first or largest term is called the order of the equation. In the Polynomial regression program, you specify the order of the equation and input points on the curve to be approximated.

Such approximations are used for the scientific functions in most of the math packages for computers. Just for fun, I let Extended BASIC compute the SINE of angles from 0 to 1.6 radians in increments of 0.1 radian, and entered these coordinates for a run of the Polynomial Regression program. I was surprised to find that a 7th order approximation produced an error that was always lost in the 8th digit of the result. The worst case error is less than 4x10©-8. This approximation involves 7 multiplications and 8 additions, so it is reasonably fast. For the SINE and COSINE functions, the angle can be reduced to a value between zero and PI/2 (about 1.59 radians).

The more straightforward way to calculate the SINE, is as a sum of some of the terms of an infinite series. The series for the sine is X-x©3/(3*2)+x©5/(5*4*3*2)-x©7/(7*6*5*4*3*2) etc. Each term is the previous term with the numerator multiplied by -(x©2) and the denominator multiplied by the next two integers. Math students will be aware that 7*6*5*4*3*2*1 may be abbreviated 7! or 7 factorial. Calculating the first 7 terms of this series requires 7 multiplications and additions and the error approaches 20% at the limit of 1.59 radians. The "approximation" is thus about a million times more accurate than a short portion of the "correct" calculation for the value. This means, of course that many more terms of the infinite series must be included to get the accuracy required to match the "approximation". For your interest, the Scientific Functions in PASCAL that were published in this column previously have been redone and just those functions are given here. The coefficients are given to 9 digits since that is the limit of Lucidata PASCAL. The 9 digit coefficients produce results that have a worst case error of less than 4 parts in the 8th digit. It was of some interest to note that the error curve is symmetrical, and that the number of peaks or maxima in the error curve corresponds to the order of the approximation polynomial. It was of interest to note that the values for the odd power coefficients of the approximation are near the values of the coefficients of the infinite series. They are somewhat "distorted" and the even power coefficients are very small for the first few terms. The reason that we can get such improved accuracy is that the infinite series in theory works for all values of angle, but the approximation only works for angles between 0 and 90 degrees.

I also found a much better approximation for the Arctengent function than I had previously used. That these approximations get better and better as the Order of the approximation is increased is logical. This fact was somewhat obscured when I ran the Regression program with 6 or 9 digit BASIC. The

program computes a "standard error of estimate", which reached a minimum and then increased with the order of the approximation equation. With the more precise arithmetic, I have seen no degredation of the error for equations of order as high as 10, though the approximations have been adequate with less terms. I find that the Arctangent requires an order 9 approximation to be about twice as bad as the order 7 approximation for the SINE. Also included here are the listings for the approximations for LOG and EXP done in this way, having reduced their range first in a manner similar to that of reducing the angle for the Trigonometric functions.

One more note regarding the Polynomial Regression program. In order to try various orders of polynomials for the approximation, it is necessary to run the program several times. I don't like typing in 12 digit values for the 17 points of input data, so I changed the input statement to a read statement, and included the values as DATA statements at the end of the program. Also, in order to try the approximations of higher order, it is necessary to increase the dimensions for some of the array variables as explained in the program listing in Borchers and Poole. I later got tired of typing in the data points even once, and modified the program to read a data file, and then wrote very short programs to generate the data points. It was interesting to note that the TSC extended BASIC is accurate enough to allow calculation of the error curve for errors in the 8th digit. TSC must be accurate to at least 12 digits for these functions. I don't have tables accurate to that many places to find out and my calculator is only good to 9 places.

SD BASIC

I finally broke down and bought the Software Dynamics BASIC compiler. As a result of the column for July "soapbox speech" that I made regarding compilers in general from a user's point of view, I received a letter from Ron Whitas of SD, informing me of their version 1.4 that allows long variable names and labels. Now they tell me!! Anyway, there were included a few programs that they had done to demonstrate the new compiler. Sorry Ron, but I think long labels can be way overdone. They produce almost as unreadable a program as do the varaible names A1 through Z9 in Basic. Further, it really gets frustrating when you have to type a very long variable name or label more than a few times.

In his sample program Ron used some of these for variable names; THEREEXISTSAWAYOUT, RANDOMDIRECTIONSAVE, and RANDOMLENGTHSAVE. Maybe with lots of care and lots of white space around statements these work for you, but I would prefer abbreviated versions such as WAYOUT, RDIRECTION, and RLENGTH. I really don't think the longer versions convey any more information except perhaps on the first reading of the program. These variables are incidentally from a program to generate a maze. It was modified by SD from the program by Paul Wennberg in Kilobaud Microcomputing, Nov. 1979. Version 1.4 also has provision for passing parameters to subroutines and functions, which make it look like Fortran or Pascal.

Aside from these observations, SD BASIC is fast, and I will soon have a series of Benchmark tests of most all of the compilers and interpreters currently available for FLEX. These will compare not only run time but memory efficiency. My current impression is that the SD compiler uses about half again as much memory as Lucidata Pascal when running the equivalent program. SD does have full scientific functions included in the runtime package, and Lucidata Pascal does not. These estimates include the Scientific Functions as I have prepared them to run with Pascal. There will be more here about the subject when I have had a chance to do the comparisons accurately.

Please excuse my previous reference to UCSD™ Pascal as USCD. It was not a typographical error, but a genuine mistake on my part. I hadn't realized that the acronym stands for University of California at San Diego.

Editor's Note:

All references to the symbol '@' should be changed to '↑'. This is a fault of our daisy wheel in the 15 character configuration.

The routines indicated above will be included in next months issue. Also some additional routines, furnished by Ron Anderson, and not published yet, will also be included.

As you can tell by this issue, we have increased the size of 68 Micro Journal. This will allow me to include more listings and other material. This is all due to the fine response our advertisers have received from YOU OUR READERS.

Also will use this space to note that we are running 'slooooow' on the 'GIANT SOFTWARE CONTEST' wrap-up. We hope to have it finished soon, but when we got to the final part of judging the entries, we found that some of the disk, especially those from overseas were 'bombed'. We hope we have them all in by now and can get it completed soon. Thanks again to each and every one of you who entered. Needless to say it was a much bigger job than we had thought.

DMW

A BATTERY BACKUP CLK BOARD

For those users who have need of a battery backed up, clock for the S50 bus, the ROBERTSON ELECTRONICS CLK 68-1 clock-interrupt board is now available from Robertson Electronics, 1003 Warm Sands Drive SE, Albuquerque, New Mexico 87123.

The CLK 68-1 is installed on the 30 pin bus and obtains all voltages from this bus, both operation and battery charge. This means that if all power is lost the clock remains active. Not just for a day or week or so, but for up to four (4) months. That should just about survive all power failures!

PHYSICAL DESCRIPTION

The clock is a combination calendar and clock board. Also available is a interrupt generator, with intervals from 488 microseconds to 256 seconds. No jumpers are required as dip switches are used for setting all features. The NI-Cad batteries furnished are in a charging state when the computer is running. This should insure dependable battery back up under normal useage. Provision are for day of week, month, day of month and time (12/24 hour format) the year is set in software. Gold pin connectors and 2 MHz operation are available as an option. The board is solder masked and silk screened. and also has available an extra parallel I/O port. It comes with all parts including the batteries and IC sockets. The advertised price is $89.95 kit and $119.95 assambled and tested.

Also as options all software on SSB disk $14.95 - cassette (Kansas City) $12.95. The 36 page manual can be ordered for $10.00 - refundable with order of CLK 68-1.

The control crystal has a trimmer for tight calibration. The board has no 'after-thought' jumpers and the board used for the review worked without a hitch.

DOCU ENTATION

The manual is well done. It consist of 36 pages of assembely instructions, applications notes and examples, checkout, calibration, theory of operation, software description, software listings and component identification charts. For an item of this type the documentation is above average, and the computer user with little construction experience, should not experience much difficulty. The only part that might give trouble is tight calibration of the crystal circuit. The standard accuracy of the crystal is .0003 percent. For those without adequate lab equipment to do a close crystal alignment or those who desire the tightest ratio

possible, the factory will do a final alignment for the modest sum of $5.95.

As stated earlier the manual is well put together and is laid out in such a manner that it is easy and logical to follow, even for a beginner. The software listings furnished in the manual are written assuming the computer is a Smoke Signal system running DOS 5.XXC or later. However, it is easy enough for the user to convert to any other system or monitor I/O running on a S50 bus machine. For those who heavily use BASIC, sample routines are given also.

OPERATION

Interfacing and addressing the board is straight forward for both calendar/clock and parallel port. Access is fast, on the order of a millisecond or so. A sample of the printout is:

FRI AUG 8 1980 (DATE)
21:45:22 (TIME 24 HR)
7:35:12 A.M. (TIME 12 HR)

We find that program listing that have the date and time printed on each one are easier to use, from the standpoint of knowing when each was written or modified. A system calendar and clock is a worthwhile addition to any computer system.

CONCLUSION

As detailed earlier the Robertson calendar/clock board is a product of excellent quality, both in design and construction. Additional information may be secured from:

Robertson Electronics
1003 Warms Sands Dr SE
Albuquerque, NM 87123
(505) 294-0025

UCSD PASCAL™

Dale Puckett
14753 Endsley
Woodbridge, VA 22193

This review deals with one of the most powerful and least understood software products available to 6809 users.

UCSD PASCAL™ for the 6809 is sold by the Microsystems Division of Control Systems, Inc. (CSI), 1317 Central Ave., Kansas City, Kansas 66102. CSI's free telephone number is 800-255-4411.

UCSD PASCAL™ is supplied to SWTPC system owners on three disks in either the 8-inch or 5 1/4 inch format. The disks and a manual written by SofTech, Inc., sell for $419. The first disk contains the operating system, a compiler, a screen editor, a filer, a linker, and a library file. A 6809 unique Interpreter and Basic Input Output System (BIOS) round out the disk.

Another disk includes a BASIC compiler, a line oriented editor for use on teletypes and other hard copy terminals, a disassembler and a calculator. The final disk contains Assemblers for both 6800 and 6809 systems.

Before pursuing the details of this very complex system we should tip our hat to Dave Allen at CSI. Dave goes out of his way to help his customers master UCSD PASCAL™.

INTRODUCTION

When you type "U" or "D" for the first time to bootstrap the system, you will find yourself in a new world. UCSD PASCAL™ is unlike anything available to 6800/6809 users before. Its forte is its transportability and machine-independence. The same system, except for the machine unique interpreter and BIOS, has been installed on the 8080, Z-80, 6502, TI 9900, 8086, NOVA, PDP-11, WD-11, Z-8000 and 68000 microprocessors. More than 15,000 copies are in existence and users have written software ranging from FORTRAN compilers and text-processors to accounting and small business packages. The idea here is that you can run their programs on your 6809 system without the strain of translating from one dialect to another, etc.

This transportability is made possible by a Pseudo-Machine architecture. The compiler generates p-code which looks like machine code to the pseudo-machine. This p-code is the same for all machines. The interpeter translates it to machine code for the system in use.

OPERATING SYSTEM

The operating system is on a file called SYSTEM.PASCAL and is the first thing you see after booting. After a nice welcome message, a prompt line is printed along the top of the screen. If your screen is 80 characters wide, you see the entire word, for example, F(iler. If it is smaller you will only see the first letter of the command because terminals with shorter lines receive one-letter prompts.

All commands are one-letter. For example to run a program, you type R. To compile a program, you type C.

Most of the disk work is done by a program called the Filer. Its one-letter commands allow you to move files, remove them, and list their directories, etc. It also keeps track of the date and keeps a record of the names of your various peripherals and their files.

COMMUNICATION

Every peripheral on your system is identified by a number and a name. You may use either. For example if you want to print a file, you transfer it to the printer. When the filer prompts you with "To where?", you may answer "PRINTER:" or "#7:". The colon tells the system that the word is a volume designator.

SWTPC DMAF drives are assigned to volumes 4 and 5. The MF-68 mini-floppy system appears as volumes 9 and 10. If you had a three (three mini 5" disk are maximum) drive minidisk system it would appear as 9, 10 and 11. Volumn #12 is used for the SWTPC CDS-1 Marksman winchester hard-disk drive.

The PRINTER mentioned earlier is assumed to be a serial interface on port 7. The drivers supplied in BIOS are dumb drivers and assume a standard ASCII terminal. To interface parallel printers, etc., you must write your own assembly language procedure and link it to the system using the Linker.

The standard UCSD package places the console on port 1 and a remote input and output on port 2. Port #0 is also used as a remote port.

GETTING IT RUNNING

Disks used on the 6809 system must be initialized using the TSC NEWDISK utility. CSI

plans to supply an initialization utility written in PASCAL in the future, however.

MEMORY

You must have 56 K of memory in your system to run UCSD PASCAL™. The operating system checks to see that 56 K is available and will return to S-BUG-E if it is not.

The package comes ready to run on a SWTPC CT-82. If you have this terminal you have it made, if not, don't worry. CSI supplies two utilities, SETUP and BINDER, which allow you to describe your terminal to the PASCAL system.

SETUP handles details such as the screen width, number of lines on the screen, backspace character, delete character, etc., much like the TTYSET utility which comes with FLEX. SETUP is completely self-prompting and is a dream to use. You can answer the prompts with a decimal number, a hexidecimal number, an octal number, or you can type the actual character you are defining. It even confirms the change and prompts again, just in case you want to change your mind. When you are finished, you may update the parameters in memory or on a disk file. If you pick the disk option, SETUP creates a file called NEW.MISCINFO which you later are invited to convert to SYSTEM.MISCINFO.

BINDER allows you to insert your own GOTOXY routines into the SYSTEM.PASCAL file. These routines tell terminals like the CT-82 where the cursor should be on the screen at all times.

Believe it or not, I used the example GOTOXY routine for extremely dumb terminals in the system manual, and it made my ancient CT-1024 appear to be smart. Unbelievable.

A CLOSER LOOK

Immediately after booting you are offered a choice of all the commands at the outermost level. The prompt line looks like this.

Command: E(dit, R(un, F(ile, C(omp, L(ink, X(ecute, A(ssem, D(ebug, ? (II.0). Typing an E will load the editor into memory and automatically load in the current workfile. The workfile is a concept central to the structure of UCSD PASCAL™. It is a scratch-pad area which can be used for the development of programs. Only one workfile is allowed at a time although any workfile may be saved elsewhere with a different filename.

The workfiles are called SYSTEM.WRK.TEXT and SYSTEM.WRK.CODE. If you type C, the system compiler will be read into memory and the workfile will be compiled if it is present. If it is not present, the compiler will prompt for a filename.

If it detects an error in the syntax of your program, the compiler will stop and display an error number. It also points to the offending text.

At this point you can type escape to abort and return to the command level, or you may type an E and go straight to the Editor. When you do this, you will find that the Editor will place the cursor near the mistake. If the error is unimportant however, you may hit the spacebar and go on with the compilation.

Typing an R will cause the current workfile to be executed. If the program has not been compiled yet, R will load in the compiler and compile it. It will also automatically call the linker if necessary. If the codefile is already existent however, it will execute it immediately.

Typing X will generate a prompt for a filename. If the filename you provide is on line, it will be executed. If not, an error message is generated.

From this level you can also call in the Linker, Assembler, or Debugger. Typing an F in the outer level will call in the Filer which will immediately print its own prompt line.

Filer: G(et, S(ave, W(hat, N(ew, L(dir, R(em, T(rans, D(ate, and Q(uit (A).

If you type a "?" you will be shown more commands. The second group allows you to check for Bad blocks, Krunch the disk, Make a new disk, Set the name of the volume Prefix, or Zero a new disk.

An overview of filer commands is provided here. G, for Get, will load the named file into the workfile. One filename here will load both the .text and .code files if they are on line.

An S will allow you to Save the file you have been working on. It prompts you for a filename.

Typing N clears the workfile and creates a blank, unnamed workfile. If there is a workfile already present you will be asked if you want to throw it away.

A W will identify the name and present state of the workfile. A V will give you a list of the Volumes you have on-line. L will produce a List of a disk directory and the system will prompt you for the volume you wish to list. C will allow you to Change the name of a volume or a particular file. R will remove a file from the directory and T will allow you to copy one file to another. It should be noted here that you can transfer a file to the CONSOLE: or PRINTER:. It took me a long time to figure out that this was the easy way to list a file. At first I loaded the file into the editor and listed it.

D will allow you to change the current Date, or check it and P will allow you to change the volume Prefix.

X, for eXamine allows you to physicaly attempt to recover suspected bad blocks on a disk. K will Krunch a disk. This means that all files present are moved so that the empty space left on the disk is contiguous.

One good point about the operating system is that system codefiles do not have to be on the system disk. The system scans all volumes that are on-line for the System files and remembers where they are.

This is extremely important to persons using the 5 1/4 inch disks. Especially, those with only 35 tracks.

The reason is that PASCAL system files are very long. For example, SYSTEM.COMPILER is 68 blocks long. The editor is 45 blocks long. A block is the equivalent of two FLEX sectors. In other words, it takes over half a disk for just the compiler.

Since the operating system scans, I am able to place the compiler, editor, linker and library all on one disk ready for use. I can then use the disk which was booted in to hold quite large workfiles. The only system files I include on it are BIOS, the SYSTEM.INTERP and the SYSTEM.FILER.

THE EDITOR

The SYSTEM.EDITOR is the principal tool for

creating, reading, and changing text files. When i first comes up it prints a prompt line on the top o the screen and begins printing the file on th second line. As much of the file that will fit i placed on the screen. The rest can be called in a will by moving the cursor.

The prompt line almost explains itself.

Edit: A(djust C(py D(lete F(ind I(nsrt J(mp R(plac Q(uit X(chng and Z(ap.

If a workfile already exists, it automatically read in when the editor is called. I it is not present you may name a file to be read, o you may hit return and start a new workfile.

A summary of the various commands availabl with the editor follows.

A, for Adjust, allows you to change th indentation of the line which contains the cursor.

C, for Copy, enables you to copy text fro another disk file into your workfile. It als allows you to move selected portions of the curren workfile into a special buffer for holding.

D, for Delete, allows you to delete characters words or any combination of the two. When you cal the function, the location of the cursor becomes th anchor. You then move the cursor at will an charecters between the two positions are remove when you type an ASCII ETX.

F, for Find, allows you to find a string in th workfile. It has two modes, Literal and Token. T explain, a Literal search for /F-STOP/ would fin only that exact character combination. A Toke search on the other hand would also find /F - STOP or /F- STOP/. It would even find the token if th two parts were on different lines.

I, for Insert, inserts text starting at th location of the cursor. It allows the use of th backspace and delete characters to reject selecte parts of the insertion.

J, for Jump, gives you a way to move the curso from one part of the file to another. You can Jum to the Beginning or to the End of the file as we as to Markers you are allowed to establish.

M, for Margin, will adjust anything locate between two blank lines to the margins you set.

P, for Page, will move the cursor one page at time. A Page here is defined as the amount of te that can be held by the screen.

Q, for Quit, will let you leave the editor. the process you may, Update the workfile; Ex without changing the workfile; Write to a nam file; or Return to the editor.

R, for Replace, lets you change one string another. It will operate in the Literal or Tok modes. And, will allow you to Verify the chan before it happens if desired.

S, for Set, gives you a chance to set Marke by assigning a name to them. It also allows you to define the Environment.

V, for Verify, will redisplay the screen with the cursor centered.

X, for eXchange, will exchange the current text in the file with the text typed. It only works in one line. A backspace will cause the original character to re-appear.

Z, for Zap, remembers the location of the last thing found, replaced, or inserted; and deletes everything between there and the current cursor position.

The SYSTEM.EDITOR supplied with the UCSD PASCAL™ package from CSI allows you to work only with files that will fit in available memory. This appears to be just over 14,000 characters and could be a disadvantage if you are working with extremely long source files. I understand, however, that UCSD is working on another editor called the L2 Editor which can edit files as large as the space available on the disk. I'm sure CSI will make it available when the UCSD folks, perfect it.

As we mentioned earlier the CSI package also includes an editor called YALOE for Yet Another Line Oriented Editor. I tried it and didn't like it. But, it does allow you to use the UCSD system with a hard terminal such as a teletype.

The SYSTEM.COMPILER is an adaptation of the portable Pascal-P compiler developed at the Swiss Technical University. It has been changed to run on microcomputers with restricted memory capacity and to handle UCSD extensions.

The compiler makes one pass through the source file thus allowing it to run relatively quickly. The cost is a relatively large amount of memory space. However, the 56K SWTPC system can handle quite large Pascal programs.

If you do have extremely large source files, this package has the answer. It allows you to operate the compiler as if it were two separate programs. Only one part is brought into memory at a time. This expands the compilers working memory by over 5 K but it does slow the process down a bit.

Another feature of the compiler which should be mentioned here is the include directive. This feature allows you to keep small parts of a Pascal program on separately edited text files.

This comes in handy when you want to use one section of program statements in several different programs. The include directive tells the compiler to regard the entire text contained in a named text file as if it were part of the source program text. It looks like this.

```
(*$I EXAMPLE.TEXT*)
```

This statment would cause the text file EXAMPLE.TEXT to be read in and compiled as if it were part of the source file being compiled. You can even insert an include file which contains CONST, TYPE, VAR, PROCEDURE and FUNCTION declarations. To do this you must use the include directive between the last variable declared in the VAR list and the first PROCEDURE or FUNCTION heading declared in the main program. This placement tells the compiler to relax the strict Pascal syntax requirement that CONST declarations occur before TYPE declarations.

DISK FILES

Another important area that should be covered in this review is the use of disk files. UCSD PASCAL™ has been extended to allow the handling of random access disk files.

The UCSD system regards all disk files as if they were composed of blocks 512 bytes long. This is made possible because of the low level routines in the BIOS package. This allows the physical characteristics of the users particular disk system to remain transparent. These details are of no concern to most programmers.

Packing of the records into the correct number of 512-byte blocks is done automatically by the Pascal operating system. This means the programmer need only request access to a specific logical record in his file by using its number.

A type definition for a typical Pascal date file might look like this.

```
TYPE INFOREC=(PERSON);
  THISREC=
    RECORD
     NAME:STRING[32];
     ADDRESS:STRING[20];
     TOWN:STRING[30];
     PHONE:PACKED ARRAY[0..9] OF CHAR;
     AMOUNT:INTEGER[9];
    END;
```

File handling reserved words include: GET, PUT, READ, WRITE, RESET, REWRITE, CLOSE, LOCK, SEEK, and EOF.

LONG INTEGERS

One of the complaints you see in some of the trade publications about UCSD Pascal is the lack of the precision of its REAL number types. But, again the designers have planned ahead. Users who need extended number lengths with complete accuracy for business and scientific applications may use the LONG INTEGER type. LONG INTEGERS are declared by using the standard INTEGER identifier followed by a length enclosed in square brackets.

```
VAR X: INTEGER[12];
```

The largest number inside the square brackets is limited to 36.

And "long" constants can be defined in a normal manner.

```
CONST LOTOFNUMBERS = 12345678987654321;
```

The example above would default to a LONG INTEGER type.

CONCLUSION

The CSI 6809 implementation of UCSD PASCAL™ works and it works well. The support from the company is very good.

This review only touches the surface of the features available in the package. There is just too much system and not enough writer or space in the Journal to do it justice.

The only problem I faced was the fact that my PTA disk driver board is not totally compatible with the SWTPC DC-2 board. Because of this the system locks up on booting when it goes over to look for a disk on volume #10. Dave Allen gave me the location of some of the disk drivers and I got the nerve to experiment. Without his help I probably would have thrown up my hands and quit. As it was I found that I could unplug the cable running to my drives and trick the system into continuing. After this everything seemed to work perfect.

This personal support from the man behind the implementation is a very important plus for the CSI package. Dave realizes that the system is so complex that, unless you are another Einstein, you can't help but have questions. And, he is happy to answer them for you.

Another plus for the system is its speed. Because of the larger 512 byte file blocks, and because of the sequential access which puts everything in logical order, disk access seems to be very fast and compilation is amazingly fast. Executing or Running a program that has already been compiled appears to be an instantaneous operation.

I personally compiled nearly a dozen Pascal programs of varying length from several magazines and text books. They all ran without a hitch.

My only complaint is about the documentation supplied by Softech, the company that holds the license from UCSD. The manual is a graphic disaster and the style is befitting a typical government bureaucrat.

On the other hand, the 6809 unique documentation which you receive from CSI is well written and attractively printed. It is easy to understand and makes life much simpler.

There are two things I would like to see with the system. A pre-packaged set of drivers for a parallel printer and a warm start address for the main system. The drivers would save me the work of writing my own. The warm start would allow re-initializing the system without rebooting when a disk hardware problem occurs. It would save the 30 plus seconds required to re-boot the system.

UCSD Pascal will be a welcome addition to your 6809 system. In the long run your initial investment will hold your software costs down. This savings will result because of the portability of the system and the large number of users who are already using it. Their programs will make your life much simpler.

Editor's Notes: The L2 Editor above mentioned is now a standard part of the CSI-2 package. For those who have purchased this package; a 'FREE' update is available merely by returning the original disk. The update will be returned with the L2 Editor and any other updates since you purchased your original version.

It should be noted that the current policy of CSI is to furnish 'FREE' updates on return of the original disk. Few software vendors go to this degree of customer support!

Additionally it should be noted that there are several 'USER GROUP' programs that WILL NOT execute in 56K. They use too much memory for variable storage; also the stack being at the top and the heap at the bottom also require memory, therefore, in some instances 56K is just not enough.

The argument for a 'Warm Start' entry point is valid; however, UCSD did not make any provision for this feature. Due to the structure of PASCAL very little time would be saved with a warm start entry point (actually less that 10 seconds) so no support at this time is available for any other entry point then 'Cold Start'.

The necessity for parallel printer routines is another need that is being filled (so I am told by Dave Allen). Dave reports that several users are working on this and it will be published in the users group newsletter (and 68 Micro Journal) as soon as it becomes available.

Any bad memory that your machine might have that has been hard to find or unable to find by normal testing routines, will certainly show up using PASCAL. PASCAL, by virtue of the 'stack' and 'heap' utilization is an effective 'memory test'!

A final note for those who find that 'booting' PASCAL is a sometimes affair. Seems that many 1771 disk controller chips are not up to snuff. The good ones seem to be the 1771B001 - 1771B02 types. So if you cannot get it to boot or it is intermittent: check your memory and 1771.

As more of you start using this version of PASCAL please keep me informed of your progress and especially any procedures or routines that you feel will help the rest of us.

"UCSD PASCAL is a trademark of the Board of Regents of the Univ. of Calif.

DMW

ADDITIONAL PASCAL INFO

For those of you who have called or written concerning Lucidata PASCAL for the 6809, I cannot report anything as we have not received a 6809 version from Lucidata for review or evaluation. If and when we do I will get a review or report published.

DMW

NOTES 2: BUSINESS PROGRAMS

By: RICHARD G. CAGLE
Applevalley Day School
11103 Sagepark Ln
Houston, TX, 77089
713-481-3586 (after 6 pm)

This article is a sequel to one published in the June issue. I received several letters (one from a college pricipal who liked the article in spite of the misspelt words).

For this issue, we will get into some specifics of business programming techniques in basic.

Since these articles are oriented to the first time business user of the 6800/6809, some of it may be very elementary to the experienced user.

INTERACTIVE COMMUNICATION

The communication between you and your computer should be in a form that both of you understand. Since your normal language is not binary or hex, then English is the only choice. Programs should have enough information for a semi-unskilled operator to make intelligent judgements. As a simple example:

```
1010 INPUT "DATE",D1
```

In this case,the operator could answer:

JULY 12,1980
or 7/12/80
or 12-jul-80
or 7,12,80
or 71280

Only the last one would be acceptable since it is the only numaric answer. TSC Extended basic would issue an error on the first 3 answers.The fourth answer is particularly bad,as it would result in just '7' being entered, since the comma signals the end of the input. SWTPC basics will ask for a re-enter for the first one, and would swallow only the leading numaric on the next three.

The solution is to give the operator just a little more info, for example:

```
1010 PRINT "PROVIDE DATE"
1011 INPUT"(USE MODAYR FORM)",D1
```

Or,if your operator is a little dumb:

```
1010 PRINT"PROVIDE DATE "
1011 PRINT"(USE 71280 FOR JUL 12 '80)"
1012 INPUT"  ",D1
```

Notice that the input area has been lined up with the example date.

The advantage of using a numeric date input is that it can be used in subsequent calculations, or can be checked for reasonableness, example: 1015 IF D1>123200 OR D1<10100 THEN 1010 This would cause a repeat of the date prompt if the date was not in this reasonable range.

In some cases there will be no need to check the date. A string input will suffice if the only use is to to print the date on a report.

If you are using TSC's Extended Basic, there is a DATE$ command which can be used to retrieve the date from the DOS. For example D$=DATE$ will give you a D$ string in the format 'DD-MON-YR'.

You may have noticed in the above examples that I have deviated from the normal practice of evenly spacing line line numbers. I do this to make my programs look ugly and to simplify debug. If the program bombed at line 1011, the '1011' tells me it is a part of a sequence that begins at 1010. A program can always be made pretty after the debugging is done using a renumberer, such as the ones sold by STAR-KITS or other advertisers.

Getting back to interactive communication, words should be used liberally to avoid operator confusion. W.L.King recently sent me his payroll program for my opinion. He was very proud of his program, and with good reason as it did many things well. Since he ran the programs himself and was completely familiar with it, he had no need for detailed operator prompting. I'm sure that he will not mind if I use his program as an example. This is what the operator saw:

```
EMP  1001 MARY JONES
DAYS WORKED? 5
EXTRA LABOR? 200
SALES? 0
CORRECTIONS? 0
```

This payroll is for a sales and service operation. They pay a daily salary plus commissions on dollars billed for sales and dollars billed for service.

The employee is not really needed, unless there are more people than the operator can keep track of. For this application the DAYS WORKED is satisfactory (although a lot of programs, including my own, use a hours worked quantity). The program does have a reasonableness check to see if more then 7 days were input. The EXTRA LABOR category does not tell what units (dollars,cents,hours,ergs) should be inputted. EXTRA LABOR(TOTAL DOLLARS BILLED) would be better. The computer display should specify the units that the program needs.

Keep in mind that it is illegal to have the operator have to whip out the old 4 function calculator. After all, part of the justification for getting the computer to help in your business was to avoid those dreary hours spent with the little red eyed monster.

The SALES? prompt is pretty obviously intended as a total dollar volume, or is it thousands of dollars? Or, maybe it is the amount of the sale commission. SALES(TOTAL $ SOLD) would be clearer.

And finally, the CORRECTIONS? prompt. This could be asking for how many mistakes were made during the week in order for the program to dock the employees pay. Assuming you understood that the prompt was asking if there were any errors in the prior entries, your normal response would be YES or Y, NO or N. It didn't work that way! The program is looking for either 0 for no corrections required or for a non-zero number if they are required. There are two things wrong with this one word prompt. First, the computer should never prompt for corrections without first displaying what it has swallowed. In the above example, I would clear the screen (or skip lines if in scroll mode) and set a display:

```
  BASED ON YOUR INPUTS:
    5.00 DAYS WORKED = $500.00 SALARY
$200.00 EXTRA LABOR =  $30.00  COMM.
  $0.00  SALES      =   $0.00  COMM.
        TOTAL WAGES = $530.00
ANY CORRECTIONS (Y=YES)?
```

This gives the operator a chance to see what was entered and the result of the computer calculations. This gives the operator the responsibility of performing a reasonableness check of the data. Of course a check can be placed in the program to warn the operator if some limit has been exceeded but at the current rate of inflation, this check would have to be redone every year! Note that all of the money amounts are shown with dollar prefix and with two decimal places shown. This is done with "DIGITS=2" in SWTPC basics or with PRINT USING in TSC Extended basic. Note also the new corrections prompt. It clearly tells that if you hit "Y" you will get a chance to make corrections, otherwise for any other character struck, you will proceed with the program. This was the second thing wrong with the one word prompt - it did not allow a default option or indicate the type of response desired. The new program logic would be:

```
INPUT"ANY CORRECTIONS (Y=YES)",A$
IF LEFT$(A$,1)="Y" THEN 1300
```

which will return to the line 1300 initial entry program if the answer is anything beginning in "Y". Or the same routine in TSC Extended would be:

```
PRINT"ANY CORRECTIONS (Y=YES)"
IF INCH$(0)="Y" THEN 1300
```

The advantage of the INCH$(0) command is that no carriage return is needed.

While we are discussing yes-no prompting, there is an important point: A smoothly flowing program should do most of the work, with the operator hitting the big fat 'RETURN' key from time to time signalling 'thats right, baby, go-man-go!'. Therefore the logic should default to allow a 'RETURN' to be used as the normal answer. In the above example the default was a'NO' choice. I have adapted the convention of always telling in the prompting message what the non-default answer is, like the (Y=YES) in the above. The words that make up the prompt should also be compatible. For example:

```
IS THIS WRONG (N=NO)
ARE THERE ERRORS(N=NO)
```

In both of these prompts there is some double negative ambiguity. The following would be preferred:

```
IS THIS RIGHT(N=NO)
```

In a few cases, you may find that a particular routine is critical, and may want to force the operator to deliberately make a choice. This can occur when there is no means to make corrections subsequent to the step and the results are final. Use something like this:

```
WARNING!
   LAST OPPORTUNITY FOR CORRECTIONS
   IS THIS CORRECT (Y=YES OR N=NO)?
```

And the program logic would check for either a 'Y' or 'N' and restate the warning if neither were inputted.

In the same category as yes-no prompts are the program pause prompts. These are used when the computer puts a lot of data on the screen,and time must be given for the operator to look at it before continuing, although no action is needed immediately. Flex has a Pause feature, but I prefer to leave it disabled and to program in pauses where they belong in the program. Most of my programs use a simple:

```
1410 INPUT "READY",A$
```

with no logic to check the answer A$. However, my son managed to interpret this as a yes-no prompt and was surprised when the program continued, even when he keyed in 'NO' to say he was not ready. In newer programs, I am using:

```
 1410 INPUT"(HIT RETURN TO CONTINUE)",A$
Or, for TSC Extended:
 1410 PRINT"(PRESS ANY KEY TO CONTINUE)"
 1411 AN$=INCH$(0)
```

A pause prompt would typically be followed by a screen clear, if in page mode. This can be done in either of two ways. One would be to set up a subroutine:

```
   20 PRINT CHR$(16),CHR$(22):RETURN
```

and use 'GOSUB 20' anytime a screen clear was wanted. Or, early in the program a variable can be defined:

```
   20 C$=CHR$(16)+CHR$(22)
```

and anytime a screen clear is desired then 'PRINT C$' will do it.

The use of page mode or scroll mode is a personal opinion choice,but I prefer the page mode, primarily because with my stock SWTPC CT-64 terminal the results are much more predictable. Occasionally, when I am displaying data that slightly exceeds one page, I will use control characters (issued by the program) to transfer to scroll mode, then back to page mode.

Here is a technique for crowding more data onto the screen during input. The input will cause a line-feed/return but you may want to print something on the same line. This example is part of an option routine in my payroll program for calculating time worked by entering clock times for IN and OUT off of the time sheet.

```
INPUT"IN",A$
(processing of A$ goes on here)
PRINT TAB(20);CHR$(11);"OUT";
INPUT B$
(B$ is processed and C$ calculated)
PRINT TAB(33);CHR$(11);"EQUALS ";C$
```

The CHR$(11) is the vertical tab on my terminal,and the above example allows two inputs and the results to be put on a single line, allowing one weeks worth of time to be displayed on a single page. TSC Extended does have a '!NPUTO' command that inhibits the '?' prompt and the C/R-L/F, however it will print the next print statement on top of the inputted data.

In most business programs there is a need to round off dollars to two significant places. The standard routine is:

```
A=INT(A*100+.5)/100
```

This routine is not needed in TSC Extended because the PRINT USING statement will perform the rounding:

```
PRINT USING 'TOTAL WAGE $$###.#',A
```

Another interesting feature of TSC Extended print using is the ability to use a variable to define the print using field. For example:

```
E$="TOTAL WAGE $$###.#"
PRINT USING E$,A
```

Which accomplishes the same thing as the previous example. Using this in a subroutine can save a few bytes of code.

TAPE-BASED BUSINESS PROGRAMS

In spite of my negative discussion of tape based business systems in the last article, I did get some enquiries about buying cassette programs. I do not have any to sell or to even give away! Yes, I did use tape at one time, and I still have the programs, but I do not consider them useable!!! Tape users should refer to a series of articles by Ron Harvey in Kilobaud, in Oct '77, Nov '77, and Dec '77. In addition to providing a useable payroll program, he also explains the techniques to use to get data on and off of the tape.

That is all for this issue. The next sequel will be several months from now. If you either need information about some particular aspect, or have a favorite algorythym or routine that you would like to share with 68' readers, please write. We will probably discuss the use of TSC's Extended basic random files, as I have already received one request for help.

STYLOGRAPH WORD PROCESSING SYSTEM

Dr. James Pomerantz
Department of Psychology
SUNY/Buffalo
Buffalo, New York 14226

The STYLOGRAPH word processing system fills a well recognized need for SS-50 bus systems. (STYLOGRAPH was named STYLUS but because of a trademark conflict the name has been changed.) It is available from Sonex Systems, Box 238, Williamsville, NY 14221. STYLOGRAPH is presently the only fully interactive text processing package available for SS-50 users. It is available only for the 6809. For those 6800 users who have been considering upgrading to the 6809, STYLOGRAPH may well be the product to convince them that the time has finally come to convert. This was certainly the case with me; once I saw STYLOGRAPH working, I ordered a 6809 processor board straight away and made the rather simple hardware and software changes that were required. This text processing package is the match of any text processing package I have seen, including those for the S-100 bus and a number of commercial systems (many of which it quickly puts to shame.)

Much of today's word processing software has been modeled after programs developed on large scale computers. On these systems, terminals are interfaced with the main computer at low baud rates and text files are handled in a batch mode so that text editing and word processing were actually separate operations. Such is the case, for example, with TSC's excellent EDIT and PR combination.

One of the first people to realize the extra conveniences and capabilities that could be achieved with a microcomputer was Michael Shrayer, who wrote the Electric Pencil for the 8080 processor. Many other software packages have improved on Shrayer's ideas but the general concept remains the same - namely that the full text should be visible on the screen and should immediately reflect changes as they are made from the keyboard. Furthermore, the editing and processing should exist as one complete package. STYLOGRAPH was clearly designed with these ideals in mind. While STYLOGRAPH contains many advances over the Electric Pencil, the overall straightforward design and ease of use, for which the Electric Pencil is well known, are retained.

Commands

There are actually three modes of operation while using STYLOGRAPH: the "ESCAPE", "INSERT", and "SUPERVISOR" modes. In the ESCAPE mode, you can scroll up and down through the text and perform all of the editing functions. In the INSERT mode, you can type in the text as you would on a typewriter. The SUPERVISOR mode is for interacting with the disk operating system and for outputing to the printer.

All of the ESCAPE commands require only a single keystroke and the keys involved are arranged logically on the keyboard to make them easy to remember. For example, all of the cursor and scrolling keys are located under the right hand, as so:

```
U I O
J K L
M , .
```

The five cursor control keys [IJKL,] are arranged in a cross. Typing an "I" moves the cursor up, an "L" moves it one space to the right, and so on. The center "K" key is an express key that moves the cursor alternately to the far left and right of the screen. The scrolling keys [UOM.] cause the screen to scroll up and down, with the "UM" keys scrolling one line and the "O." keys scrolling a full page. The cursor stays in its position while scrolling. If you try to scroll off the screen the cursor will automatically move to the next line. Similarly, if you try to move the cursor off the screen, STYLOGRAPH will scroll to keep the cursor on the screen.

The usual complement of search, replace, copy, block move and delete commands are available and fully implimented. They are all quite simple to use but I will explain the use of the replace command so you can see the care with which STYLOGRAPH was designed. After you hit the "R" key, an area is cleared out on the top of the screen and the rest of the text appears below this area. In the top of the screen two messages will appear asking you for the string you want replaced and the string you want to replace it with. After you have entered these, the cursor will then move to the next occurrence of this string and ask you if you want the string replaced or not. Simply answer "Y" or "N". The screen will immediately reflect the change and ask you if you want to go on to the next string. You may, if you are brave, answer "A", indicating that you want all the occourences of the string changed without being prompted on each one. The importance of all of this is that you always see exactly what is going on. Also, there are clearly understandable English messages which appear during the editing process.

As you are typing in a line in the INSERT mode, there is no need to watch for the end of the line since STYLOGRAPH will take a word that overflows off the end of the line, insert it on the next line and bump all of the following text to the right [and, if necessary, down to the next line.] STYLOGRAPH automatically scrolls the text up or down if the cursor tries to move off the screen.

In either the ESCAPE or INSERT modes you may also execute a series of CONTROL commands. These commands serve a variety of functions. There are three delete commands: single character, word, and line. Whenever STYLOGRAPH encounters an error, you will hear a "bell" from your terminal. There is a control command to display the error message (again in English) and then return the screen to normal. A "page status" command causes the screen to be rewritten with status information. This includes things such as line lengths, file names, memory left, and so forth. There is a CONTROL command that causes the formatting commands to disappear from the screen. The tab functions (set, clear, and move to tab) are CONTROL commands that operate much like typewriter tab functions.

One exception to the "what you see is what you get" rule is that many advanced printer capabilities, such as boldface, underlining, superscript and so forth, simply cannot be shown on the screen. The solution to this problem is clever and fun to watch. Whenever a character has some special characteristic, it is displayed on the screen in low intensity. If you want to see what characteristic the character has, you hit the control key (control "O" indicating "overlay") which causes the character to be replaced by another character. If, for example, a word has been underlined, the word will be replaced by a string of underlines. Hitting the key again will return the screen to normal.

Disk Interfaces

STYLOGRAPH is fully interfaced to the FLEX disk operating system. You may load additional files at any point in the text and save all or portions of a text to a file. This kind of flexibility makes it especially easy to create "boilerplate" documents, as they are called. You can output any or all pages to the printer or a text file for later spooling. You may also execute most FLEX commands from STYLOGRAPH. Sonex has indicated that Microware Corporation will be marketing a version for their OS-9 disk operating system and a version for UNIFLEX will appear early next year.

Formatting

Format commands are entered as normal text while you are in the INSERT mode, very much as you would on the TSC text processing system. These formatting commands will, of course, not appear on the printed output. Separate headers and footers with page numbers and titles can be defined. Centering, right justification, full justification, indents, pagination, line lengths, spacing, vertical tabbing, left margin, page numbers, and page lengths can all be specified. Moreover, a "ghost hyphen" feature is included that allows long words to be hyphenated during formatting only if the word happens to fall at the end of a line.

One of the truly unique characteristics of STYLOGRAPH is that the text is formatted on the screen much as it will appear on the final printed output. The effects of most of the formatting commands are directly visible on the screen. In contrast to word processing packages using separate editor and processor programs, with STYLOGRAPH there is little need to get a printed output to assure that everything is formatted correctly. Headers, footers, page numbers, justification, and so forth appear directly on the screen. If you have been surprised at printout time while using a batch mode processor, you will appreciate this feature. Also, it is fun to change some parameter, such as the line length or justification, and see the entire screen reformat almost instantly before your eyes.

STYLOGRAPH supports both TTY and specialty printers. The specialty printers, such as the NEC Spinwriter, Diablo, or Qume can do letter quality printing and have a number of advanced features that STYLOGRAPH uses. If the lines are fully right and left justified, STYLOGRAPH microspaces between all of the letters to even out the line rather than insert spaces between the words, resulting in a much cleaner looking copy. A number of other printing options are also supported on this printer as the following examples, taken from the manual, will show:

operator controllable pitch

and line spacing

BOLDFACE

superscript[1], subscript$_2$

underline, overline,

or [illegible]

If you have a non-specialty printer (eg., Decwriter) and think you might move up to a specialty printer someday, I would advise buying one of the specialty printer versions instead of the TTY version since the specialty versions can drive a TTY printer but not vice versa.

Documentation

The manual that comes with STYLOGRAPH is one of the easiest to read I have encountered. It is designed so that even an inexperienced user can learn the system with little or no assistance. The new user can sit down at the terminal with the manual at hand and learn by actual use of the system. A text file of part of the manual itself is included on the disk to experiment with and to see how an actual text file looks. One minor objection that I have to the manual is that there is no index. This is only a minor quibble, since the table of contents is logically arranged and complete. A summary of the commands is included in an appendix. I would advise copying this page and keeping it near the terminal while learning the system.

Cost and Updates

This review is based upon my use of STYLOGRAPH with a Gimix 80x24 board, but I have seen it work with a CT-82 and a number of other terminals (eg., SOROC). It is designed to work with any terminal that has a clear screen function and cursor addressing and which runs at 9600 baud or better. Since terminals and printers vary so much in the way they operate, a different version of STYLOGRAPH must be purchased for each combination of printer and terminal. Fortunately, you are not left in the lurch if you buy some new peripherals since Sonex will update and replace your version with another for a modest fee. The present update fee is $30.00 plus any price difference. The current price for STYLOGRAPH is $150 for proportional spacing printer versions and $135 for tty type printer versions. This is less than half the price of similar software sold for other systems.

Conclusions

Since writing and editing text, whether it be assembler files, high-level language programs, letters, manuscripts or whatever, is the most time consuming use of most computers, any software that makes this task easier is certainly welcome. The simple, human engineered design and the full complement of formatting commands should make it particularly well suited to many office environments.

Despite STYLOGRAPH's many advanced text processing features many users will find that it will also earn its keep as an editor for BASIC and assembler text files. When you are editing, the text is always fully visible. You can always see the line you are

working on along with the adjacent lines. There is no need to relist the progam to see if you have made any errors. The ability to see the structure of the program is a great asset. The live, on-screen formatting is astonishingly fast with STYLOGRAPH, and unlike with the Electric Pencil, for example, it seems to be impossible to outtype the program; no matter what else it may be doing, it always seems to find time to accept characters typed in. It is difficult to appreciate all the virtues of cursor based editing until you try to go back to editing using a normal line editor.

For users who already have a large number of text files on hand that have been prepared for outputting with a seperate text processor program, the conversion process is fast and simple. I have converted several text files prepared with TSC's EDIT for PR with no problems whatsoever.

Since I have been singing STYLOGRAPH's praises, you must be wondering if it has any faults. There are some features it lacks that may be required by some users. It cannot handle files larger than memory, so long text files must be broken up into sections. Sonex says that this restriction was necessary to make the system simple from the user's standpoint and to make it run as fast as possible. For my own editing I normally break up files when they get too large anyway. There are also no provisions for mail-list generation. I would think that an auxiliary program of this type would be a valuable addition to STYLOGRAPH, and I would not be surprised to see it offered soon.

STYLOGRAPH is certainly a welcome addition to the extensive line of software already available for the 6800. With all of the exellent, high-level software available it is strange that such a package has not appeared sooner. In any case, 68-50 users no longer need mumble excuses or change the subject when they are asked about interactive text processing for their machines.

POSITION INDEPENDENT CODE FOR THE 6800

New developements in microprocessors promise to allow much easier and more flexible coding. One of the most interesting of the concepts being discussed is position independent coding. Position independent code will run, without modification, anywhere in the memory of a computer. This makes it much more flexible than conventional coding and simpler to combine with other code. You don't have to wait to be able to use this technique, you can use it now with the 6800.

The instruction set of the 6800 microprocessor was designed for general computing. As a result it is still the easiest CPU to program and the only CPU of its generation which can use position independent code. It's not as easy to write PIC for the 6800 as it will be for the 6809, but the 6800 is available now. You can use 6800 machines to learn the utility of this type of coding and to get ready for the next generation of microcomputers.

PIC is written using only relative addressing, this means that all memory references and all program flow controlling instructions must use addresses that are specified relative to some point in the program. When the program is loaded into memory, the absolute addresses of all parts of the program depend on its location, but the relative displacements between the parts is postion independent.

The 6800 has a limited relative addressing mode. It can only be used with the branching instructions, and only for a relative displacement of -125 to +129 bytes. This mode uses a single address byte (following the instruction byte, eg. 2509, where 25 is the branch instruction and 09 is the displacement) and the address is calculated by adding this byte to the address of the instruction following the branch. This addressing mode is very useful for controlling program flow in short programs or in segments of larger programs, but it is far from adequate for PIC. Fortunately we can use the 6800's indexed addressing mode to create a true relative addressing mode.

Indexed addressing is available on most of the instructions in the 6800 instruction set. A single byte address code is used, but the actual address is calculated by adding (as an unsigned binary number) this byte to the contents of the index register. The 6800 index register is 16 bits wide so the resulting address can be anywhere in memory. The single byte displacement allows the range from X (the index registers contents) to X+255. By changing the contents of the index register we can address any location in memory.

A position independent program can be written by first loading the index register with a known location in the program and then using indexed addressing throughout the program. The only problems are getting the known address into the index register and keeping track of its current value as the program runs. We must be able to modify the index register's contents to allow addressing more than a 256 byte range.

Lets consider the problem of getting a known location into the index register. We want to load the absolute address of a known location in the program into this register. What we really need is an instruction which loads the contents of the program counter into X. The address of this instruction would then serve as the known location on which we would base all relative addresses. We don't have a single instruction which will do this, but we can write a short section of code which acts like a transfer PC to X instruction. The first step is to get PC onto the stack. A BSR (Branch SubRoutine) instruction will push the current PC onto the stack. By using a 00 displacement on this BSR, we avoid any disruption of normal program flow. The BSR is followed by PUL A, STA A instructions which restore the stack pointer and load the PC value into memory. Finally the index register is loaded from the temporary storage location (XTEMP and XTEMP+1) in memory. This temporary storage is also useful for later manipulation of the known location.

Since XTEMP holds the high order byte of the base address, we can increment the base address in steps of 256 bytes by incrementing XTEMP. This allows us to move the base address throughout memory. This technique, in combination with the X to X+255 range of indexed addressing, allows relative addressing anywhere in the memory of the computer.

FIGURE 1 ROUTINE TO GET KNOWN LOCATION

```
        BSR ADR         PUSH PC ONTO STACK
ADR     PUL A           GET THE HIGH ORDER BYTE
        STA A XTEMP     STORE IT AT XTEMP
        PUL A           GET THE LOW ORDER BYTE
        STA A XTEMP+1   STORE IT AT XTEMP+1
        LDX XTEMP       LOAD X FROM XTEMP
```

PUTTING IT INTO PRACTICE

Lets look at some examples of coding for a position independent program. First we have the code which loads the index register with the base address. This must be the first executable code in the program. See Figure I
After this code is executed, the absolute address of ADR is stored in XTEMP and in the index register. To address a location within 255 bytes forward of ADR, we use indexed addressing with a displacement equal to the displacement between ADR and the desired address. To address locations further forward (or behind ADR) we must first change the base address stored in X.

Its easy to determine the relative addresses for any location in the program. If you assemble the program (with ADR for all of the indexed displacements) with ADR located at 0000, then the absolute addresses will be the relative displacements. Remember that the program is position independent. You can choose the starting address for ease of finding the displacements. This first assembly should be just to get these addresses. After you put them into the program (in place of the dummy values), you can reassemble it anywhere you wish.

For a program longer than 256 bytes, the base address must be modified by the program as it runs. This is necessary because of the 255 byte limit on indexed addressing. The best way to modify this address is by incrementing or decrementing XTEMP and then loading the index register with the modified base address. This modification can be performed at any point in the program, but it may become very hard to keep track of the current value of the base address. One solution to this problem is to keep the original base address in separate memory and reload XTEMP from it before each modification. Another solution is to keep all the modifications in a single section of the program. This is the approach that I have used.

Lets look at the control routine for a monitor program. This routine accepts a two character command and jumps to a subroutine determined by the command. I've used BSR's to control flow within the control routine and indexed JSR's (Jump SubRoutine) to get to the subroutines. Each subroutine returns control to the control routine when it is finished. A position independent "jump table" contains all the step which modify the base address. If this jump table is kept to less than 256 bytes, it can be located before the ADR base address. In this position changes in its length will not change the relative displacements of the remainder of the program. Figure II contains flowcharts for the control routine, the jump table, and a sample subroutine.

The command loop starts with a routine to find the base address. This is followed by an I/O routine which puts out a ? and waits for a two character input command. This command is converted to a single byte command code stored in the A accumulator. The command loop then turns control over to the jump table.

In the jump table, the command code is compared to each of the allowed codes and, if a match is found, an indexed jump to the selected subroutine is executed. If no match is found, control is returned to the command loop. Program flow is then branched back to the start of the program.

As the command code is successively compared to each possible code, the value stored in XTEMP is incremented and reloaded into X whenever the address of the next subroutine is out of range for indexed addressing with the current base address. Each incrementing moves the base address 256 bytes forward. If a very large change in the base address was required, a constant could be added to XTEMP, or subtracted from it. For changes up to 1K bytes it takes less code to use incrementing or decrementing. Part of the code for a jump table is shown in Figure III.

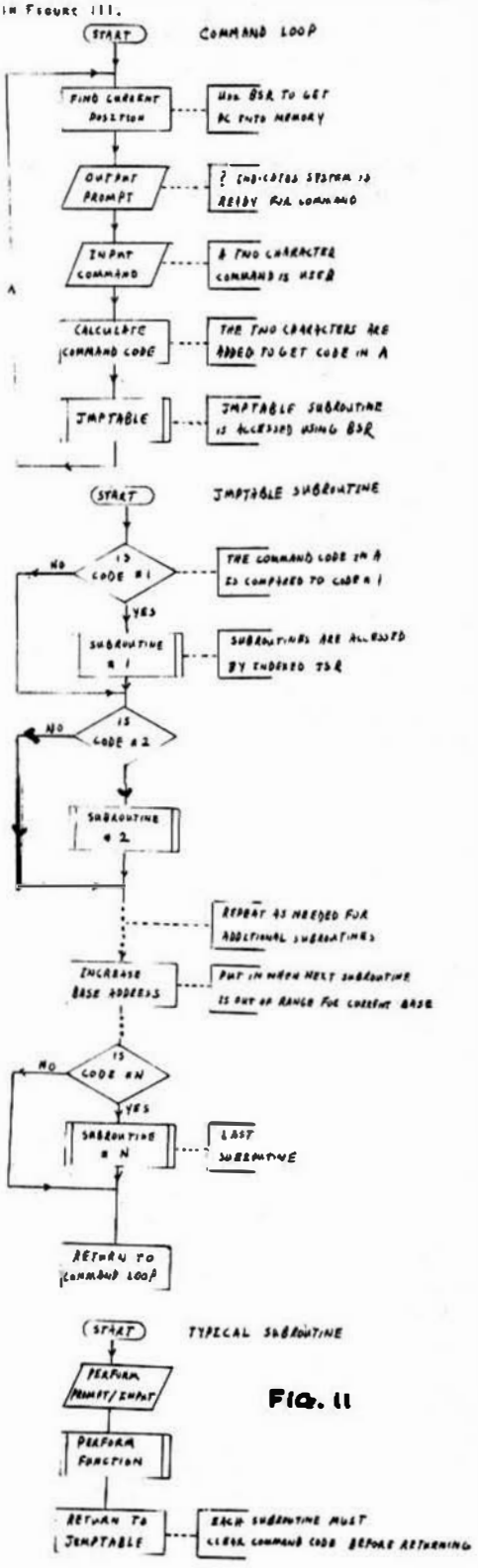


Fig. II

Each subroutine required a six byte code sequence in the jump table: compare A to the command code (two bytes), branch around the jump if not equal (two bytes), and jump to the subroutine (two bytes). This means that up to 20 command codes could be implemented in a jump table less than 125 bytes long. If the base address must be incremented, four additional bytes are required: increment XTEMP (one byte) and load X (3 bytes). Figure III shows a section of a position independent jump table.

Once the program has reached the subroutine selected by the command, the I/O routines in the computer's operating system will probably be needed. With MIKBUG or similar systems, these routines are written as subroutines so we can use regular JSR's to access them. A problem occurs with PDATA which requires that X point to the character string to be output. In this case X, which is pointing somewhere in the 256 bytes preceeding the start of the subroutine, must be changed before PDATA is used.

If the output string is not located within 256 bytes of X, we must do a two byte addition to XTEMP and XTEMP+1. When the string is less than 256 bytes from X, only XTEMP+1 needs to be changed before XTEMP is loaded into X. In either case the amount which must be added can be calculated by subtracting the current location of X from the address of the first byte of the string.

A position independent subroutine which can be used to calculate relative displacements is given below. It also illustrates some of the techniques mentioned earlier.

FIGURE III PIC JUMPTABLE

```
TABLE  CMP A #$93    COMPARE CODE TO FIRST IMPLEMENTED CODE
       BNE   T1      BRANCH AROUND JUMP IF NOT EQUAL
       JSR   $6E,X   USE INDEXED JUMP TO GET TO FIRST ROUTINE
T1     CMP A #$A9    REPEAT FOR SECOND COMMAND CODE
       BNE   T2
       JSR   $99,X   USE INDEXED JUMP TO GET TO SECOND ROUTINE

REPEAT AS NEEDED

       INC   XTEMP   INCREMENTING XTEMP INCREASES BASE ADDRESS
       LDX   XTEMP   BY $100. THEN NEW BASE IS LOADED INTO X
       CMP A #$A6    FIRST ROUTINE BEYOND NEW BASE ADDRESS
       BNE   T6
       JSR   $02,X   ROUTINE STARTS $02 BEYOND NEW BASE
T6     CMP A $8E     CONTINUE WITH OTHER COMMAND CODES

REPEAT AS NEEDED

T9     CMP A #$5E    JUMPS CAN BE USED TO GET TO THE MONITOR
       JMP   START   START IS IN MIKBUG
       RTS   RETURN  TO COMMAND LOOP
```

This program is a simple combination of two routines in a position independent monitor supplement I use. Because the routines are position independent, I was able to combine them by simply putting one after the other. The only change required was in the displacement to the prompts. This displacement was based on a known address in the command loop of the supplement. I changed it to a known position in the first routine. In the monitor supplement, the first routine, RANGE, sets the limits used by a number of other routines including BRANCH. This supplement is used with a RT 68/MX monitor, but all the I/O addresses in this example are MIKBUG compatible.

The program can be loaded using the punch listing given to run at $3F00 or, by using the code given, to run any where in memory, except for $0030 to $0035 which is used for temporary storage of variables. Before you run it, you must load $34 and $35 with the address of RA1 ($3F01 in the punch listing). In the monitor supplement, XTEMP is loaded with the base address by a BRS, PUL A, etc. You can combine this program with the base address finding routine given earlier. Just remember to change the displacement for the prompts to match the new base address.

FIGURE II BRANCH CALCULATING SUBROUTINE

```
0C             CLC                START OF SUBROUTINE
8621    RA1    LDA A  #$21        LOAD DISPLACEMENT TO "FROM"
9935           ADC A  XTEMP+1     ADD IT TO BASE ADDRESS
2403           BCC    RA2         CHECK FOR CARRY
7C0034         INC    XTEMP       INCREMENT HIGH ORDER BYTE
9735    RA2    STA A  XTEMP+1     SAVE LOW ORDER BYTE
DE34           LDX    XTEMP       LOAD FROM ADDRESS INTO X
BDE07E         JSR    PDATA1      PUT OUT FROM
DF34           STX    XTEMP       SAVE ADDRESS OF TO
BDE047         JSR    BADDR       GET FIRST ADDRESS
DF32           STX    EADD        SAVE IT
DE34           LDX    XTEMP       POINT TO TO
BDE07D         JSR    PDATA2+2          PUT IT OUT
BDE047         JSR    BADDR       GET SECOND ADDRESS
DF30           STX    BADD        SAVE IT
46             FCB    'F,'R,'O,'M,4
52
4F
4D
04
54             FCB    'T,'O,4
4F
04
9631    BR1    LDA A  BADD+1      TWO BYTE SUBTRACTION
9233           SBC A  EADD+1      USING THE ADDRESSES
9735           STA A  XTEMP+1     JUST ENTERED
9630           LDA A  BADD
9232           SBC A  EADD        THE RESULT IS STORED IN
9734           STA A  XTEMP       XTEMP
8620           LDA A  #'          LOAD A WITH A SPACE
BDE1D1         JSR    OUTEEE      PUT IT OUT
CE0034         LDX    #XTEMP      POINT TO XTEMP
BDE0C8         JSR    OUT4HS      PUT OUT ITS CONTENTS
4F             CLR A              CLEAR COMMAND CODE
39             RTS                RETURN TO CALLING PROGRAM

S11E3F000C8621993524037E003497350E34BDE07EDF34BDE047DF320E34BD07A
S11E3F1BDE07DBDE047DF30465244F4D04544F04C963192339735963097329703
S1113F3634862 0BDE1D1CE0034BDE0C84F3941
S9
```

CURRENT APPLICATIONS AND FUTURE POSSIBILITIES

Position independent code for the 6800 is harder to write and slightly longer than conventional code. This limits its usefulness to programs which really benefit from the freedom to be run anywhere in memory. Two types of programs fit this category:

PROGRAMS WHICH ARE TO DISTRIBUTED TO A LARGE NUMBER OF COMPUTERS WHICH MAY BE CONFIGURED DIFFERENTLY, AND PROGRAMS WHICH ARE DESIGNED AS PART OF A LIBRARY OF PROGRAMS TO BE USED INDEPENDENTLY OR IN COMBINATIONS WITH OTHER PROGRAMS.

A GENERAL PURPOSE MONITOR PROGRAM OR A PROGRAM DESIGNED TO SUPPLEMENT AN EXISTING MONITOR (EG MIKBUG) COULD REACH A WIDER GROUP OF USERS IF IT COULD BE RUN, WITHOUT MODIFICATION, WHERE THE USER HAS AVAILABLE MEMORY. DISK BOOT STRAP PROGRAMS ARE ANOTHER GOOD EXAMPLE OF THE BENEFITS OF PIC. WHY SHOULD YOU HAVE TO ADD ROM IN A SPECIFIC LOCATION WHEN YOU MAY ALREADY HAVE AVAILABLE ROM IN ANOTHER LOCATION. THESE TYPES OF PROGRAMS ARE SHORT ENOUGH AND SIMPLE ENOUGH TO BE WRITTEN TO RUN ANYWHERE. ON THE OTHER HAND, A LARGE, COMPLICATED, PROGRAM LIKE A HIGH LEVEL LANGUAGE COMPILER WOULD BE VERY DIFFICULT TO WRITE WITH PIC AND DOES NOT REALLY GAIN ANYTHING FROM BEING POSITION INDEPENDENT.

LIBRARY ROUTINES, WHICH ARE COMMON IN LARGE COMPUTERS, CAN MAKE COMPLICATED SCIENTIFIC OR BUSINESS PROGRAMMING MUCH EASIER. THESE PROGRAMS CAN BE CALLED IN TO MEMORY AND RUN WHEN THEIR FUNCTIONS ARE NEEDED BY A MAIN OR CONTROL PROGRAM. PIC ALLOWS MUCH SIMPLER IMPLEMENTATION OF LIBRARY FUNCTIONS SINCE THE ROUTINES CAN BE RUN (IF THEY WILL FIT) IN WHATEVER MEMORY HAPPENS TO BE CURRENTLY AVAILABLE. IT ALSO ALLOWS LARGE ROUTINES TO BE BUILT UP FROM A SERIES OF LIBRARY ROUTINES AND RUN ON SIMPLER SYSTEMS WHICH DO NOT HAVE THE FAST MASS STORAGE REQUIRED TO TAKE ADVANTAGE OF A RUN TIME LIBRARY.

WHEN THE NEXT GENERATION OF CPUS BECOME READILY AVAILABLE, MUCH OF THE CODE WRITTEN FOR THEM WILL BE POSITION INDEPENDENT. THEY WILL HAVE ADDRESSING MODES SPECIALLY DESIGNED TO MAKE PIC EASY TO CODE AND FAST TO RUN. BUT, AS I SAID AT THE BEGINNING OF THIS ARTICLE, IF YOU HAVE A 6800 SYSTEM YOU DON'T HAVE TO WAIT FOR THOSE NEW CPUS. WHY NOT EXPLORE THE POSSIBILITIES OF PIC NOW, WITH YOUR PRESENT SYSTEM.

M. BOYD
ST. MARY OF THE PLAINS
DODGE CITY, KS 67801

MORSRX

Terry A. Mayhugh W4MSR
11632 Midhurst Dr.
Concord, TN 37922

MORSRX is a Morse Code receiving program for the SWTP 6800 microcomputer. In addition to the computer the only hardware required to demonstrate the program is a telegraph key and an audio oscillator. Since the decoding algorithms are implemented in software some rather elaborate features are easily included. The program is self-adapting to code speed variations allowing solid copy over a wide range of speeds without operator adjustment. Parenthesis parity, backspace on error character, and sidetone monitor control are some of the features of this software controlled decoder. MORSRX is ORG'ed at location $0900 and since it is less than 1 kilobyte long it will easily fit in a minimally configured system. An optional interface is described which enables the user to interface any SSB receiver to the computer so that code can be taken directly off the air. In this mode of operation a software filtering routine effectively filters out static and other atmospheric noise to prevent erratic copy.

Software Description

Although a simplified flow diagram of MORSRX appears in Figure 1 many of its salient features are pointed out in the comments accompanying the assembled listing. When executed, the program configures the I/O and then waits for the space bar to be depressed before continuing. Then when key-down is sensed the computer turns on the sidetone oscillator and measures the amount of time that the key is down. When key-up is sensed the sidetone is turned off, and the computer then decides whether the element just sent was a noise spike, a dot, or a dash. In order to make this decision MORSRX keeps a running time average of a dash length and updates this average with a 1:3 weight everytime a dash is decoded. The program considers any decoded pulse shorter than 1/16 of an average dash length to be noise and ignores it. This feature of MORSRX is very effective in minimizing spurious outputs due to atmospheric noise. A detected pulse longer than 1/2 the average dash length is labeled a dash while pulse widths between these two limits are counted as dots. Key-up time is measured next. When key-down occurs again the computer decides whether a word or a character has just been completed or whether it is in the middle of a character. Now the average dash length is compared with the key-up time in order to make this decision. The appropriate output actions are then taken as indicated in the flow chart. If the incoming code is too slow so that the internally initialized timer overflows then MORSRX will output the ">" character. The byte at location $0987 should be increased to a slightly greater value using the computer system monitor if this is troublesome. At present the program will handle an initial speed range from about 5 WPM to about 30 WPM. Everytime an element (dot or dash) is decoded the program checks the length of the character it is presently working on. A "#" is output if the character is illegal because of its length or if it just cannot be found in the program look-up table. If the Morse error character (8 dots) is sent, the computer will backspace to correct the last sent character. Since in Morse the left and right parenthesis characters are represented by the same elements, the program keeps track of the number of parenthesis sent and outputs them in matching pairs. Also, when the end-of-message character (.-.-.) is sent the computer outputs a carriage return, line feed sequence and then waits for additional text. The program has no latch up modes. Whenever it becomes the least bit confused due to a poor fist at the other end or a high level static crash, it tries very hard to decode the present character and then restarts to accept the next character.

Hardware Description

The input to the receiver/computer interface can be connected directly across the speaker terminals of any SSB receiver. A voltage divider consisting of R1 and the channel resistance of Q1 is the variable gain network controlled by the AGC loop. Both halves of IC1 and their associated passive components combine to form a narrow bandpass filter centered at 1KHz. This filtered output is fullwave rectified and amplified by IC2 which transforms the 1KHz AC to 2KHz for more efficient envelope detection. The rectified output is applied to the AGC error amplifier (1/2 IC3). Since

effective AGC action is necessary for proper operation of a non-saturating tone decoder, the design parameters of this loop were determined after considerable study of the effects of atmospheric and adjacent channel interference on the operation of MORSRX. A fast attack time (100 msec time constant) is set by R29-C8 while the decay time constant set by R17-C8 is lengthened to 22 secs by diode D5. The high gain of the AGC error amplifier provides the required loop gain for precision AGC control and the reference current through R15 determines the level of the stabilized output. AGC action is effective over a 40 dB dynamic range and the circuit is designed to operate over an input range from .1 to 10 volts peak. The full wave rectified output of IC2 is also filtered by R13-C10 and is applied to the other half of IC3 which is connected as a comparator with 500 mV of hysteresis. The 1.5 msec time constant of this filter efficiently filters the 2KHz AC component leaving only the envelope of the keyed CW waveform. The comparator squares up the detected pulses and they are routed through Q3 to the computer. The setting of R19 determines the detection threshold and this should normally need adjustment only once under actual operating conditions. Since key-down corresponds to a low at the collector of Q3, a telegraph key can be connected from this point to ground for program testing or manual operation. Transistor Q2 flashes an led when a CW pulse is detected and operates independently of the computer or MORSRX. The tone and volume settings of the sidetone oscillator may be adjusted as desired by R26 and R28. The oscillator itself is keyed on and off by the computer while operating under MORSRX. Since this keyed audio is indicative of what the computer is actually decoding, the SWL receiver may be fine tuned while listening to this monitor. The low power supply currents required allow the ± 12 volt and the +5 volt power to be taken directly from the EXTP 6800 computer.

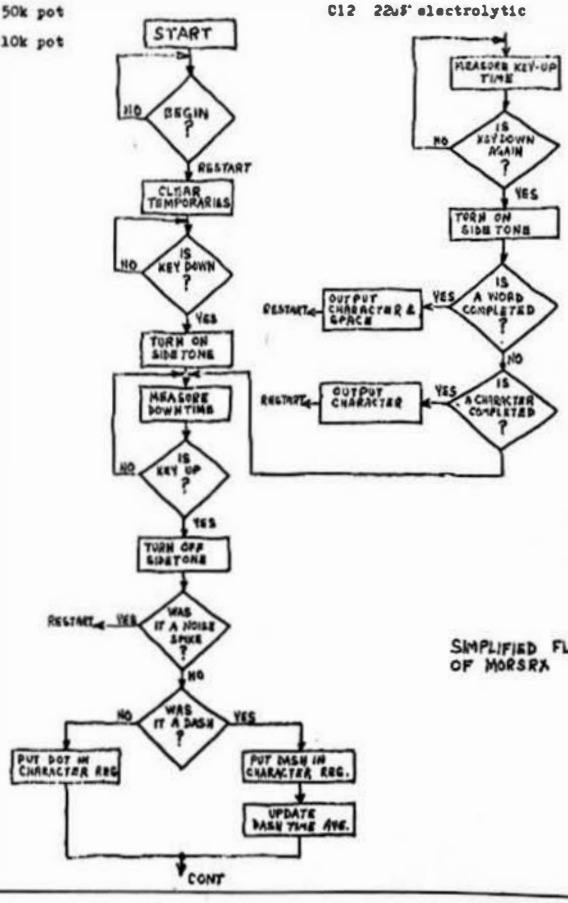
Project Checkout

The program must be typed into the computer from a keyboard starting at memory location 80900. Of course, the program should be immediately saved to tape or disk before proceeding. The A side of the EXTP PPIA (or PPI) parallel interface card should be jumper configured for computer output, and the B side should be jumpered for computer input. This card should be placed in slot #2. The LSB of the A side should be connected to diode D6 of the interface and the LSB of the B side should be connected to the collector of Q3. A telegraph key should also be connected from the Q3 collector to ground. If the receiver interface is not constructed, a key can just be connected to the LSB of the B side of the parallel card. A jump to location 80910 should be performed using the system monitor. The banner "MORSRX READY...PRESS SPACE BAR TO BEGIN RECEIVING" should appear. Press the space bar and, using the telegraph key, send code into the computer: and the text will be displayed on the terminal. To terminate MORSRX, the computer reset must be pressed. Subsequently typing 0 (under MIKBUG, SWTBUG, etc.) will restart the program. To check out the receiver interface connect an SWL receiver to the input of the interface and type 0 to bring up MORSRX. Tune in a moderately strong CW station, and adjust the receiver tuning until the led flashes in synch with the keyed 1KHz tones from the receiver speaker. If the led does not flash, the detection threshold should be lowered by adjusting R19. If adjusted too low, the led will remain on continuously. Once this coarse adjustment is made, press the space bar and adjust the tone and volume levels on the sidetone monitor to comfortable levels. The receiver should now be fine tuned for the best reception as heard through the MORSRX sidetone monitor. R19 should now be finally adjusted for best copy while listening to the monitor and should not require further adjustment. Reset the computer and restart MORSRX to begin receiving code. Copy will not be letter perfect with some stations due to static crashes and high level adjacent channel interference. Also, since some fists are colored with personal dialects, misplaced spaces and erroneous characters will may appear.

Parts List MORSRX INTERFACE

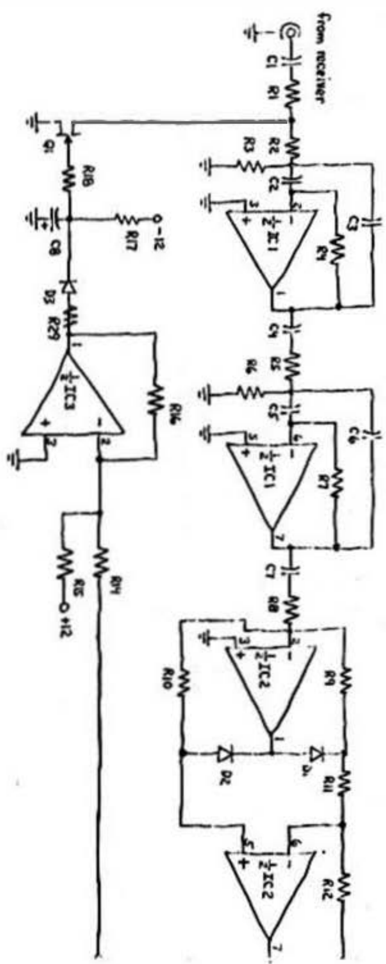
R1, R15 5.1k 10% carbon
R2, R5 12.1k 1% metal film
R3, R6 316 ohm 1% metal film
R4, R7 121k 1% metal film
R8 470 ohm 10% carbon
R9,R10,R11,R12,R13, 10k 1% metal film
R14, R18, R24, R27 1k 10% carbon
R16 470k 10% carbon
R17 4.7 Meg 10% carbon
R19 10k pot
R20 47k 10% carbon
R21, R23 10k 10% carbon
R22 270 ohm 10% carbon
R25, R26 10% carbon
R26 50k pot
R28 10k pot

D1-D6 1N914 or equivalent
D7-led
Q1 2N4858 Nchan JFET
Q2-Q4 2N2222
IC1-IC3 Dual 741 op AMP
IC4 555 timer
SP1 miniature 2-50 ohm speaker
misc. telegraph key, enclosure
C1,C2,C3,C4,C5,C6 .027uf 5% mylar
C7 1uf non-polarized
C8 47uf solid tantalum
C9 47uf electrolytic
C10 .15uf mylar
C11 .22uf mylar
C12 22uf electrolytic

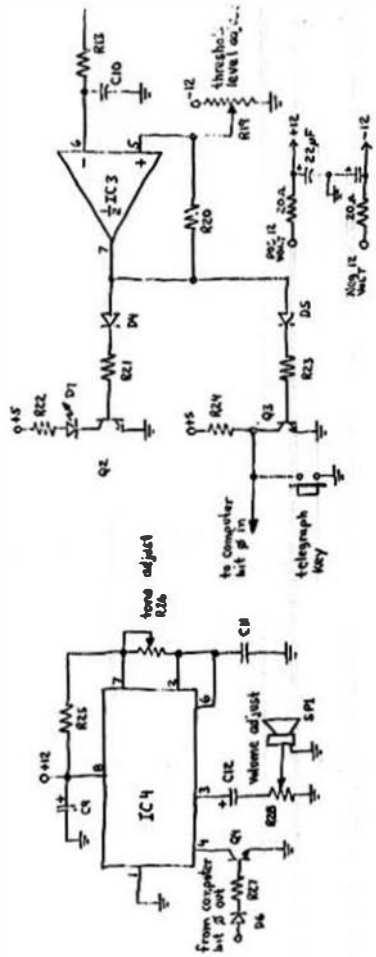


SIMPLIFIED FLOWCHART OF MORSRX

```
                   NAM    MORSRX
           ************************************************************
           *                                                          *
           * MORSRX...A MORSE C DE RE EIVING PROGRAM FOR THE 6800 MICRO *
           *                TERRY L. MAYHUGH (W4MSR)                  *
           *                                                          *
           ************************************************************
8008            PIA1AD EQU   $8008
8009            PIA1AC EQU   $8009
800A            PIA1BD EQU   $800A         ASSUMES PIA AT PORT 2 IN SWTP 6800
800B            PIA1BC EQU   $800B         (MODIFY PORT NUMBER AS REQUIRED)
A048            PROGCO EQU   $A048
A042            STACK  EQU   $A042
E1D1            OUTEEE EQU   $E1D1
E07E            PDATA1 EQU   $E07E
E1AC            INEEE  EQU   $E1AC
0008            BAKSPC EQU   $08
0900                   ORG   $0900
0900            UPTIME RMB   1             KEY-UP TIME OF PRESENT E EMENT
0901            DWNTIM RMB   2             KEY-DOWN TIME OF PRESENT ELEMENT
0903            CHAREG RMB   1             PRESENT CHARACTER WHILE BEING BUILT
0904            LENGTH RMB   1             LENGTH OF PRESENT CHARACTER
0905            DSHTIM RMB   2             RUNNING TIME AVG OF A DASH
0907 03         TIMEOS FCB   $03           MAJOR TIMING LOOP CONSTANT
0908 EE         TIMINS FCB   $EE           MINOR TIMING LOOP CONSTANT
  09            PPAR   RMB   1             PARITY REGISTER FOR PARENTHESIS
090A 10         HOMSTR FCB   $10,$16,$0D,$00,$00,$04
090B 16 0D
090D 00 00
090F 04

0910 CE 0910    START  LDX   #START
0913 FF A048           STX   PROGCO
0916 8E A042           LDS   #STACK

                *   STRUCTURE B SIDE OF PIA FOR LSR INPUT

0919 86 FE             LDA A #%11111110
091B B7 800A           STA A PIA1BD
091E 86 01             LDA A #%00000001
0920 B7 8008           STA A PIA1AD
0923 86 04             LDA A #%00000100
0925 B7 800B           STA A PIA1BC
0928 B7 8009           STA A PIA1AC

                *   INITIALIZE DATA REGISTERS AND TEMPORARIES

092B BD 0A0F           JSR   TONOFF
092E 86 39             LDA A #$39
0930 B7 0905           STA A DSHTIM        INITIALIZE FOR APPROX 15 WPM
0933 BD 0A02           JSR   CLEAR
0936 CE 0C4E           LDX   #BANNER
0939 BD E07E           JSR   PDATA1
093C BD E1AC    RESTRT JSR   INEEE         BEGIN RECEIVE WHEN ANY KEY IS PRESSED
093F 7F 0909           CLR   PPAR
0942 BD 0A0F    NUCHAR JSR   TONOFF
0945 7F 0900           CLR   UPTIME
0948 7F 0901           CLR   DWNTIM
094B 7F 0903           CLR   CHAREG
094E 7F 0904           CLR   LENGTH
0951 BD 0A15    WAIT   JSR   GOGETT
0954 26 FB             BNE   WAIT          WAIT FOR KEYDOWN

                *   KEY IS DOWN!

0956 BD 0A09    STILDN JSR   TONON
0959 BD 09D2           JSR   TIMER
095C 7C 0901           INC   DWNTIM        BEGIN TIMING THE DOWN-TIME
095F B6 0901           LDA A DWNTIM
0962 26 07             BNE   CONTI
0964 86 3E      REDUCE LDA A #'>           WARNING THAT RX SPEED IS SET TOO FAST
0966 BD 0C4A           JSR   OUTPUT        TIMEOS SHOULD BE INCREMENTED
0969 20 D7             BRA   NUCHAR
096B BD 0A15    CONTI  JSR   GOGETT
096E 27 E6             BEQ   STILDN        CONT UNTIL KEY IS UP

                *   KEY IS UP!

0970 BD 0A0F           JSR   TONOFF
0973 F6 0905    FILTER LDA B DSHTIM        FILTER OUT STATIC CRASHES
0976 54                LSR B
0977 54                LSR B
0978 54                LSR B
0979 54                LSR B               NOW HAVE DSHTIM/16
 97A F1 0901           CMP B DWNTIM
097D 22 C3             BHI   NUCHAR        ELEMENT TOO SHORT...START OVER
097F F6 0905           LDA B DSHTIM
0982 54                LSR B               CMP AVG DASH L TH/2 WITH KEY DWN TIME
0983 F1 0901           CMP B DWNTIM        WAS IT A DASH OR A DOT?
0986 22 18             BHI   INDOT         IT WAS A DOT--ADD IT TO CHAREG
0988 BD 0A1B           JSR   UPDA E        IT WAS A DASH--UPDATE RX SPEED
098B 0D         INDASH SEC
098C 79 0903           ROL   CHAREG
098F 0D                SEC
0990 79 0904           ROL   LENGTH
0993 F6 0904           LDA B LENGTH
0996 C4 40             AND B #%01000000
0998 27 1B             BEQ   STILUP        CHARACTER IS LEGAL...CONT.
099A BD 0AB5           JSR   TOOLNG
099D 7E 0942           JMP   NUCHAR        CHARACTER IS ILLEGAL...START OVER
09A0 0C         INDOT  CLC
09A1 79 0903           ROL   CHAREG
09A4 0D                SEC
09A5 79 0904           ROL   LENGTH
0 A8 F6 0904           LDA B LENGTH
09AB C4 80             AND B #%10000000
09AD 27 06             BEQ   STILUP        SO FAR CHARACTER IS LEGAL
09AF BD 0A74           JSR   ERROR         THIS ONE MAY NOT BE--CHECK IT
09B2 7E 0942           JMP   NUCHAR

                *   KEY IS STILL UP!

0 B5 8D 1B      STILUP BSR   TIMER         FINALLY START MEASURING KEY-UP TIME
09B7 7C 0900           INC   UPTIME
09BA B6 0900           LDA A UPTIME
09BD 27 22             BEQ   ENDWRD
09BF 8D 54             BSR   GOGETT
```



```
09C1 26 F2    2161          BNE   BTILUP

              2181 *   KEY IS DOWN AGAIN!

09C3 BD 0A09  1201          JSR   TONON
09C6 F6 0905  1211          LDA B DSHTIM
09C9 0C       1221          CLC
09CA 59       1231          ROL B
09CB F1 0900  1241          CMP B UPTIME      IS (DASH LEN TH)X2 > KEY UP-TIME?
09CE 22 1B    1251          BHI   ENDCHR      IF YES THEN WORD NOT ENDED
09D0 20 0F    1261          BRA   ENDWRD
09D2 B6 0907  1271 TIMER    LDA A TIMEOS
09D5 F6 0908  1281          LDA B TIMIWS
09D8 4A       1291 LPOUTS   DEC A
09D9 26 01    1301          BNE   LPINS
09DB 39       1311          RTS
 9DC 5A       1321 LPINS    DEC B
09DD 27 F9    1331          BEQ   LPOUTS
09DF 20 FB    1341          BRA   LPINS
09E1 BD 5C    1351 ENDWRD   BSR   OUTCHR      A WORD IS COMPLETED
09E3 86 20    1361          LDA A #$20
09E5 BD E1D1  1371          JSR   OUTEEE
09E8 7E 0942  1381          JMP   NUCHAR      SET UP FOR NEXT CHARACTER
09EB F6 0905  1391 ENDCHR   LDA B DSHTIM      IS CHARACTER IS COMPLETE?
09EE 54       1401          LSR B
09EF F1 0900  1411          CMP B UPTIME      IS (DASH LENGTH)/2 > KEY UP-TIME?
09F2 22 05    1421          BHI   MIDCHR      IF YES THEN CHARACTER NOT COMPLETED
09F4 BD 49    1431          BSR   OUTCHR      CHARACTER IS COMPLETED
09F6 7E 0942  1441          JMP   NUCHAR
09F9 7F 0901  1451 MIDCHR   CLR   DWNTIM      STILL IN MIDDLE OF PRES CHAR
09FC 7F 0900  1461          CLR   UPTIME
09FF 7E 0956  1471          JMP   STILDN
0A02 CE 090A  1481 CLEAR    LDX   #HOMSTR
0A05 BD E07E  1491          JSR   PDATA1
0A08 39       1501          RTS
0A09 86 00    1511 TONON    LDA A #$00        TURN ON SIDETONE
0A0B B7 8008  1521          STA A PIA1AD
0A0E 39       1531          RTS
0A0F 86 01    1541 TONOFF   LDA A #$01        TURN OFF SIDETONE
0A11 B7 8008  1551          STA A PIA1AD
0A14 39       1561          RTS
0A15 B6 800A  1571 GOGETT   LDA A PIA1BD
0A18 84 01    1581          AND A #$01
0A1A 39       1591          RTS
0A1B 7F 0906  1601 UPDATE   CLR   DSHTIM+1    (MSB)
0A1E 7F 0902  1611          CLR   DWNTIM+1    (MSB)
0A21 5F       1621          CLR B
0A22 B6 0905  1631          LDA A DSHTIM
0A25 78 0905  1641          ASL   DSHTIM
0A28 79 0906  1651          ROL   DSHTIM+1
0A2B BB 0905  1661          ADD A DSHTIM      A-> 3 X DSHTIM  (LSB)
0A2E F9 0906  1671          ADC B DSHTIM+1    B-> 3 X DSHTIM (MSB)
0A31 BB 0901  1681          DD  A DWNTIM      A-> 3 X DSHTIM + DWNTIM (LSB)
0A34 F9 0902  1691          ADC B DWNTIM+1    B-> 3 X DSHTIM + DWNTIM (MSB)
0A37 54       1701          LSR B
0A38 46       1711          ROR A
0A39 54       1721          LSR B
0A3A 46       1731          ROR A             DIVIDE BY 4 TO GET 3:1 WEIGHTED AVG
0A3B B7  905  1741          STA A DSHTIM      PUT NEW UPDATED BYTE BACK
0A3E 39       1751          RTS
              1761 *   CHARACTER JUMP VECTORS
0A3F B6 0904  17 1   TCHR   LDA A LENGTH
0A42 81 01    1781 EL1      CMP A #%00000001
0A44 26 02    1791          BNE   EL2
0A46 20 42    1801          BRA   ELMEN1
0A48 81 03    1811 EL2      CMP A #%000 0011
0A4A 26 02    1821          BNE   EL3
0A4C 20 4D    1831          BRA   ELMEN2
0A4E 81 07    1841 EL3      CMP A #%00000111
0A50 26 02    1851          BNE   EL4
0A52 20 6A    1861          BRA   ELMEN3
0A54 81 0F    1871 EL4      CMP A #%00001111
0A56 26 03    1881          BNE   EL5
0A58 7E 0B05  1891          JMP   ELMEN4
0A5B 81 1F    1901 EL5      CMP A #%00011111
0A5D 26 03    1911          BNE   EL6
0A5F 7E 0B79  1921          JMP   ELMEN5
0A62 81 3F    1931 EL6      CMP A #%00111111
0A64 26 03    1941          BNE   EL7
0A66 7E 0BED  1951          JMP   ELMEN6
0A69 81 7F    1961 EL7      CMP A #%01111111
0A6B 27 18    1971          BEQ   TOOLNG
0A6D 81 FF    1981 EL8      CMP A #%11111111
0A6F BD 03    1991          BSR   ERROR
0A71 7E 0942  2001          JMP   NUCHAR
0A74 B6 0903  2011 ERROR    LDA A CHAREG
0A77 81 00    2021          CMP A #%00000000
0A79 26 0A    2031          BNE   TOOLNG
0A7B B6 08    2041          LDA A #BAKSPC
0A7D BD E1D1  2051          JSR   OUTEEE
0A80 B6 08    2061          LDA A #BAKSPC
0A82 7E 0C4A  2071          JMP   OUTPUT
0A85 B6 2A    2081 TOOLNG   LDA A #'*
0A87 7E 0C4A  2091          JMP   OUTPUT

              2111 *   CHARACTER LOOK-UP TABLE

   A B6 09 3  2131 ELMEN1   LDA A CHAREG
0A8B 81 00    2141 LLTRE    CMP A #%00000000
0A8F 26 05    2151          BNE   LLTRT
0A91 B6 45    2161          LDA A #'E
0A93 7E 0C4A  2171          JMP   OUTPUT
0A96 B6 54    2181 LLTRT    LDA A #'T
0A98 7E 0C4A  2191          JMP   OUTPUT
0A9B B6 0903  2201 ELMEN2   LDA A CHAREG
0A9E 81 00    2211 LLTRI    CMP A #%00000000
0AA0 26 05    2221          BNE   LLTRN
0AA2 B6 49    2231          LDA A #'I
0AA4 7E 0C4A  2241          JMP   OUTPUT
0AA7 81 02    2251 LLTRN    CMP A #%00000010
0AA9 26 05    2261          BNE   LLTRA
0AAB B6 4E    2271          LDA A #'N
0AAD 7E 0C4A  2281          JMP   OUTPUT
0AB0 81 01    2291 LLTRA    CMP A #%00000001
0AB2 26 05    2301          BNE   LLTRM
0AB4 B6 41    2311          LDA A #'A
0AB6 7E 0C4A  2321          JMP   OUTPUT
0AB9 B6 4D    2331 LLTRM    LDA A #'M
0ABB 7E 0C4A  2341          JMP   OUTPUT
0ABE B6 0903  2351 ELMEN3   LDA A CHAREG
```




```
0AC1 81 00    2361 LLTRB    CMP A #%0000 000
0AC3 26 05    2371          NE    LLTRU
0AC5 B6 53    2381          LDA A #'S
0AC7 7E 0C4A  2391          JMP   OUTPUT
0ACA 81 01    2401 LLTRU    CMP A #%00000001
0ACC 26 05    2411          BNE   LLTRW
0ACE B6 55    2421          LDA A #'U
0AD0 7E 0C4A  2431          JMP   OUTPUT
0AD3 81 03    2441 LLTRW    CMP A #%00000011
0AD5 26 05    2451          BNE   LLTRR
0AD7 B6 57    2461          LDA A #'W
0AD9 7E 0C4A  2471          JMP   OUTPUT
0ADC 81 02    2481 LLTRR    CMP A #%00000010
0ADE 26 05    2491          BNE   LLTRG
0AE0 B6 52    2501          LDA A #'R
0AE2 7E 0C4A  2511          JMP   OUTPUT
0AE5 81 06    2521 LLTRG    CMP A #%00000110
0AE7 26 05    2531          BNE   LLTRD
0AE9 B6 47    2541          LDA A #'G
0AEB 7E 0C4A  2551          JMP   OUTPUT
0AEE 81 04    2561 LLTRD    CMP A #%00000100
0AF0 26 05    2571          BNE   LLTRQ
0AF2 B6 44    2581          LDA A #'D
0AF4 7E 0C4A  2591          JMP   OUTPUT
0AF7 81 07    2601 LLTRQ    CMP A #%00000111
0AF9 26 05    2611          BNE   LLTRK
0AFB B6 4F    2621          LDA A #'O
0AFD 7E 0C4A  2631          JMP   OUTPUT
0B00 B6 4B    2641 LLTRK    LDA A #'K
0B02 7E 0C4A  2651          JMP   OUTPUT
0B05 B6 0903  2661 ELMEN4   LDA A CHAREG
0B08 81 08    2671 LLTRB    CMP A #%00001000
0B0A 26 05    2681          BNE   LLTRC
0B0C B6 42    2691          LDA A #'B
0B0E 7E 0C4A  2701          JMP   OUTPUT
0B11 81 0A    2711 LLTRC    CMP A #%00001010
0B13 26 05    2721          BNE   LLTRF
0B15 B6 43    2731          LDA A #'C
0B17 7E 0C4A  2741          JMP   OUTPUT
0B1A 81 02    2751 LLTRF    CMP A #%00000010
0B1C 26 05    2761          BNE   LLTRH
0B1E B6 46    2771          LDA A #'F
0B20 7E 0C4A  2781          JMP   OUTPUT
0B23 81 00    2791 LLTRH    CMP A #%00000000
0B25 26 05    2801          BNE   LLTRJ
0B27 B6 48    2811          LDA A #'H
0B29 7E 0C4A  2821          JMP   OUTPUT
0B2C 81 07    2831 LLTRJ    CMP A #%00000111
0B2E 26 05    2841          BNE   LLTRL
0B30 B6 4A    2851          LDA A #'J
0B32 7E 0C4A  2861          JMP   OUTPUT
0B35 81 04    2871 LLTRL    CMP A #%00000100
0B37 26 05    2881          BNE   LLTRP
0B39 B6 4C    2891          LDA A #'L
```

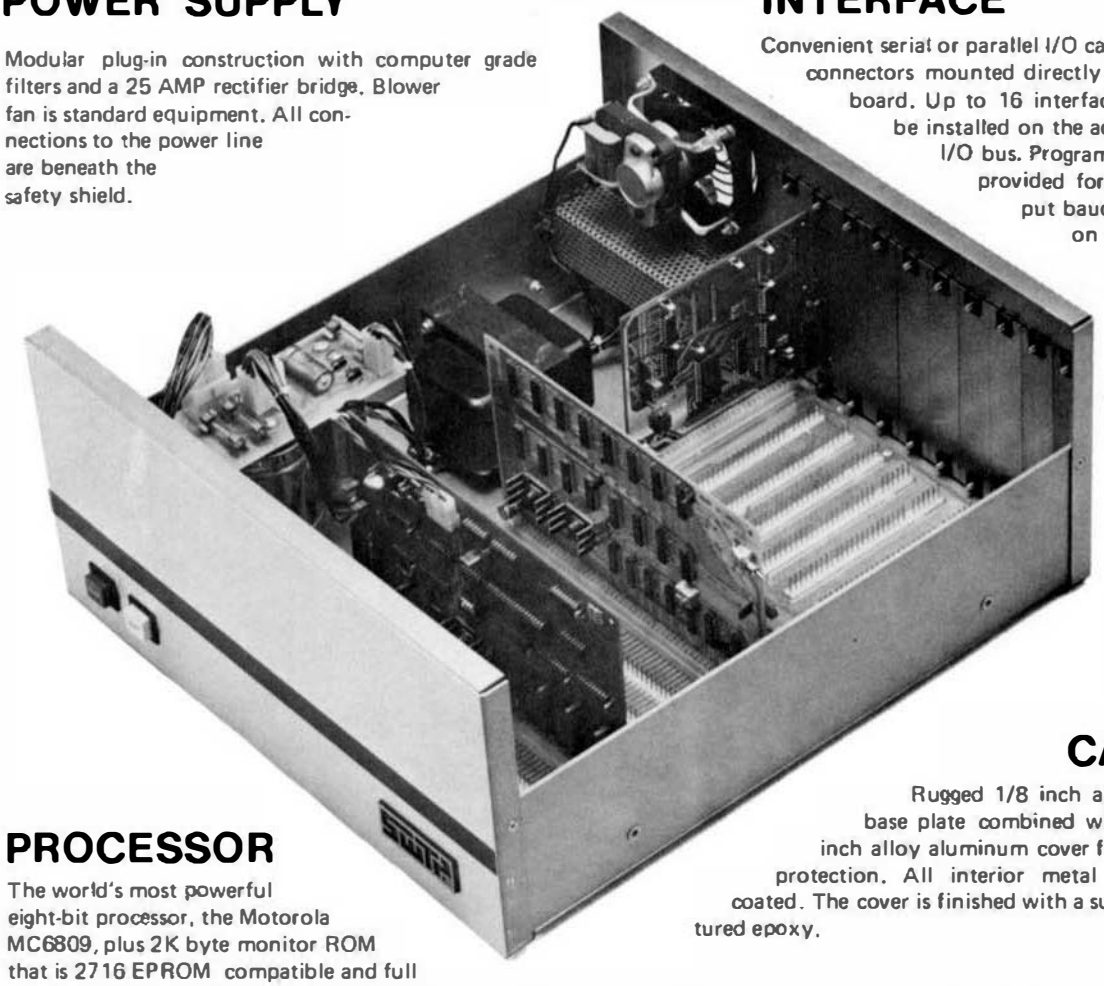
WE HAVE A 6809 FOR YOU

POWER SUPPLY

Modular plug-in construction with computer grade filters and a 25 AMP rectifier bridge. Blower fan is standard equipment. All connections to the power line are beneath the safety shield.

INTERFACE

Convenient serial or parallel I/O cards have DB-25 connectors mounted directly on the circuit board. Up to 16 interface devices may be installed on the address decoded I/O bus. Programming strips are provided for input and output baud rate selection on each port. All outputs are fully buffered.



CABINET

Rugged 1/8 inch alloy aluminum base plate combined with a solid 1/8 inch alloy aluminum cover for unsurpassed protection. All interior metal is conversion coated. The cover is finished with a super tough textured epoxy.

PROCESSOR

The world's most powerful eight-bit processor, the Motorola MC6809, plus 2K byte monitor ROM that is 2716 EPROM compatible and full buffering on all output lines. Built-in multiuser capability, just add I/O cards to operate a multi-terminal system.

MEMORY— You can purchase the computer with either 8K bytes of RAM memory (expandable to 56K), or with the full 56K. The efficient, cool running dynamic memory used in this system is designed and manufactured for us by "Motorola Memory Systems Inc."

PERIPHERALS— The wide range of peripheral hardware that is supported by the 6809 includes: dot matrix printers (both 80 and 132 column), IBM Electronic 50 typewriter, daisy wheel printers, 5-inch floppy disk system, 8-inch floppy disk systems and a 16 megabyte hard disk.

SOFTWARE— The amount of software support available for the 6809 is incredible when you consider that it was first introduced in June, 1979. In addition to the FLEX9 operating system, we have a Text Editor, Mnemonic Assembler, Debug, Sort-Merge, BASIC, Extended BASIC, MultiUser BASIC, FORTRAN, PASCAL and PILOT.

69/K Computer Kit with 8K bytes of memory . . . . . $ 495.00
69/A Assembled Computer with 8K bytes of memory . . . . . $ 595.00
69/56 Assembled Computer with 56K bytes of memory. $1,595.00

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 **(512) 344-0241**

6809 DISK SYSTEMS

All disk systems are supplied with our version of FLEX 9, the world standard disk operating system for the 6809. Our systems normally operate in double density format, but they are compatible with single density, or single sided recording formats. FLEX is supplied with over forty utilities, many of which are only available with our systems.

Our disk systems offer you mass storage at low cost. The cost per thousand bytes of storage for our various systems is shown in the chart. Other 6809 disk systems have costs up to three times greater for the same general type drive.

TYPE	CAPACITY	COST
D-5	720,000 bytes	$1.80 per/K
DT-5	1,400,000 bytes	$1.16 per/K
DMF-2	2,400,000 bytes	$1.04 per/K
CDS-1	16,000,000 bytes	$ .27 per/K

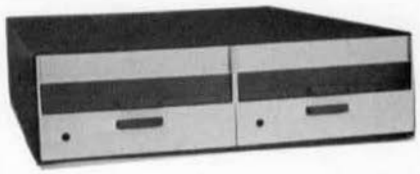
D-5 Two double sided, double density, 5" disk drives with a total on line capacity of 720,000 bytes of data. Includes cabinet, power supply, connecting cable and controller. Controller will operate up to four drives. This is an ideal disk system for small stand alone word processing systems, or for businesses that do not work with large inventories.

14 x 6 x 10 – 20 lbs $1,295.00

D-5 or DT-5

DT-5 Double track density version of the D-5. The DT-5 uses two 96 track per inch drives to provide an on line capacity of 1,400,000 bytes. Includes cabinet, power supply, connecting cable and controller. Controller will operate up to four drives. This is a disk system with enough capacity to include small inventories of up to 1,000 items, plus the usual business package of general ledger payroll, etc.

14 x 6 x 10 – 20 lbs $1,695.00



DMF2

DMF-2 Double sided, double density, dual eight-inch disk system with an on line capacity of 2,400,000 bytes. Our "top of the line" disk system features a DMA type controller for fastest possible data transfers. This drive was designed for larger businesses and multi user installations. The DMF-2 will provide the fast operation necessary for systems running multiterminals under the UniFLEX operating system. Complete with a heavy duty 1/8-inch metal cabinet, power supply, connecting cable and controller. The controller will operate up to four drives.

17½ x 5 x 21½ – 53 lbs $2,495.00

CDS-1 This "Winchester" type hard disk provides both large storage capacity and high speed operation. The CDS-1 is the answer for systems that must handle large inventories or systems with more than four terminals. The controller has its own processor and uses DMA data transfer.

CDS-1 – 115 lbs $4,395.00

CDS-1

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

```
0B3D 7E 0C4A  2901          JMP   OUTPUT
0B3E 81 06    2 11 LLTRP    CMP A #%00000110
0B40 26 03    2921          BNE   LLTRD
0B42 86 50    2931          LDA A #'P
0B44 7E 0C4A  2941          JMP   OUTPUT
0B47 81 0D    2951 LLTRD    CMP A #%00001101
0B49 26 05    2961          BNE   LLTRV
0B4B 86 51    2971          L A A #'Q
0B4D 7E 0C4A  2981          JMP   OUTPUT
0B50 81 01    2991 LLTRV    CMP A #%00000001
0B52 26 05    3001          BNE   LLTRX
0B54 86 56    3011          LDA A #'V
0B56 7E 0C4A  3021          JMP   OUTPUT
0B59 81 09    3031 LLTRX    CMP A #%00001001
0B5B 26 05    3041          BNE   LLTRY
0B5D 86 58    3051          LDA A #'X
0B5F 7E 0C4A  3061          JMP   OUTPUT
0B62 81 0B    3071 LLTRY    CMP A #%00001011
0B64 26 05    3081          BNE   LLTRZ
0B66 86 59    3091          LDA A #'Y
0B68 7E 0C4A  3101          JMP   OUTPUT
0B6B 81 0C    3111 LLTRZ    CMP A #%00001100
0B6D 26 05    3121          BNE   OHOH4
0B6F 86 5A    3131          LDA A #'Z
0B71 7E 0C4A  3141          JMP   OUTPUT
0B74 86 2A    3151 OHOH4    LDA A #'*
0B76 7E 0C4A  3161          JMP   OUTPUT
0B79 B6 0903  3171 ELMEM5   LDA A CHARED
0B7C 81 0F    3181 NNUM1    CMP A #%00001111
0B7E 26 05    3191          BNE   NNUM2
0B80 86 31    3201          LDA A #'1
0B82 7E 0C4A  3211          JMP   OUTPUT
0B85 81 07    3221 NNUM2    CMP A #%00000111
0B87 26 05    3231          BNE   NNUM3
0B89 86 32    3241          LDA A #'2
0B8B 7E 0C4A  3251          JMP   OUTPUT
0B8E 81 03    3261 NNUM3    CMP A #%00000011
0B90 26 05    3271          BNE   NNUM4
0B92 86 33    3281          LDA A #'3
0B94 7E 0C4A  3291          JMP   OUTPUT
0B97 81 01    3301 NNUM4    CMP A #%00000001
0B99 26 05    3311          BNE   NNUM5
0B9B 86 34    3321          LDA A #'4
0B9D 7E 0C4A  3331          JMP   OUTPUT
0BA0 81 00    3341 NNUM5    CMP A #%00000000
0BA2 26 05    3351          BNE   NNUM6
0BA4 86 35    3361          LDA A #'5
0BA6 7E 0C4A  3371          JMP   OUTPUT
0BA9 81 10    3381 NNUM6    CMP A #%00010000
0BAB 26 05    3391          BNE   NNUM7
0BAD 86 36    3401          LDA A #'6
0BAF 7E 0C4A  3411          JMP   OUTPUT
0BB2 81 18    3421 NNUM7    CMP A #%00011000
0BB4 26 05    3431          BNE   NNUM8
0BB6 86 37    3441          LDA A #'7
0BB8 7E 0C4A  3451          JMP   OUTPUT
0BBB 81 1C    3461 NNUM8    CMP A #%00011100
0BBD 26 05    3471          BNE   NNUM9
0BBF 86 38    3481          LDA A #'8
0BC1 7E 0C4A  3491          JMP   OUTPUT
0BC4 81 1E    3501 NNUM9    CMP A #%00011110
0BC6 26 05    3511          BNE   NNUM0
0BC8 86 39    3521          LDA A #'9
0BCA 7E 0C4A  3531          JMP   OUTPUT
0BCD 81 1F    3541 NNUM0    CMP A #%00011111
0BCF 26 05    3551          BNE   DASH
0BD1 86 30    3561          LDA A #'0
0BD3 7E 0C4A  3571          JMP   OUTPUT
0BD6 81 11    3581 DASH     CMP A #%00010001
0BD8 26 05    3591          BNE   FBARR
0BDA 86 2D    3601          LDA A #'-
0BDC 7E 0C4A  3611          JMP   OUTPUT
0BDF 81 12    3621 FBARR    CMP A #%00010010
0BE1 26 05      31          BNE   OHOH5
0BE3 86 2F    3641          LDA A #'/
0BE5 7E 0C4A  3651          JMP   OUTPUT
0BE8 86 2A    3661 OHOH5    LDA A #'*
0BEA 7E 0C4A   671          JMP   OUTPUT
0BED B6 0903  3681 ELMEM6   LDA A CHARED
0BF0 81 15    3691 PERIOD   CMP A #%00010101
0BF2 26 05    3701          BNE   COMA
0BF4 86 2E    3711          LDA A #'.
0BF6 7E 0C4A  3721          JMP   OUTPUT
0BF9 81 33    3731 COMA     CMP A #%00110011
0BFB 26 05    3741          BNE   QUEST
0BFD 86 2C    3751          LDA A #','
0BFF 7E 0C4A  3761          JMP   OUTPUT
              3771
0C02 81 0C    3781 QUEST    CMP A #%00001100
0C04 26 05    3791          BNE   COLN
0C06 86 3F    3801          LDA A #'?
0C08 7E 0C4A  3811          JMP   OUTPUT
0C0B 81 38    3821 COLN     CMP A #%00111000
0C0D 26 05    3831          BNE   SEMI
0C0F 86 3A    3841          LDA A #':
0C11 7E 0C4A  3851          JMP   OUTPUT
0C14 81 2A    3861 SEMI     CMP A #%00101010
0C16 26 05    3871          BNE   PAREN
0C18 86 3B    3881          LDA A #';
0C1A 7E 0C4A  3891          JMP   OUTPUT
0C1D 81 2D    3901 PAREN    CMP A #%00101101
0C1F 26 14    3911          BNE   ENDMSG
0C21 7C 0909  3921          INC   PPAR
0C24 F6 0909  3931          LDA B PPAR
0C27 C4 01    3941          AND B #$01
0C29 26 05    3951          BNE   LPAR
0C2B 86 29    3 01 RPAR     LDA A #')
0C2D 7E 0C4A  3971          JMP   OUTPUT
0C30 86 28    3981 LPAR     LDA A #'(
0C32 7E 0C4A  3991          JMP   OUTPUT
0C35 81 05    4001 ENDMSG   CMP A #%00000101
0C37 26 0F    4011          BNE   CHOH4
0C39 86 0D    4021          LDA A #$0D
0C3B BD E1D1  4031          JSR   OUTEEE
0C3E 86 0A    4041          LDA A #$0A
0C40 BD E1D1  4051          JSR   OUTEEE
0C43 86 16    4061          LDA A #$16
0C45 7E 0C4A  4071          JMP   OUTPUT
0C48 B6 2A    4081 CHOH4    L A A #'*
0C4A BD E1D1  4091 OUTPUT   JSR   OUTEEE
0C4D 39       4101          RTS
0C4E 4D       4111 BANNER   FCC   /MOMBAX READY ... PRESS SPACE BAR /
0C4F 4F 52
0C51 53 52
0C53 58 20
0C55 52 45
0C57 41 44
0C59 59 20
0C5B 2E 2E
0C5D 2E 20
0C5F 50 52
0C61 45 53
0C63 53 20
0C65 53 50
0C67 41 43
0C69 45 20
0C6B 42 41
0C6D 52 20
0C6F 54       4121          FCC   /TO BEGIN RECEIVING/
0C70 4F 20
0C72 42 45
0C74 47 49
0C76 4E 20
0C78 52 45
0C7A 43 45
0C7C 49 56
0C7E 49 4E
0C80 47
0C81 0A       4131          FCB   $0A,$0A,$0D,$04
0C82 0A 0D
0C84 04
              4141          END
     NO ERROR(S) DETECTED
```

Hardware Hiccup Hangs up MP-A2 EPROM

Geoffrey A. Gaas
5240 S. W. Dosch Rd.,
Portland, Oregon 97201

Owners of SWTPC MP-A2 6800 CPU's may have intermittent troubles when using one or two 2716 EPROM's with HI-PROM selected, as for DISKBUG* or home-brew monitors.

The problem is the 1k pull-down resistors R6 and R8, which come into play when the 4K-8K and 8K switches are open. The values for R6 and R8 were evidently selected when SWTPC thought they could get 74LS139 chips for IC13, and never changed when 74S139's were substituted.

A 1k is an inadequate pulldown for even standard TTL, much less Schottky circuits. Some MP-A2 boards may — if given a cool, serene, anivetless environment — play quite a while with no problems; others may turn off the monitor EPROM at various temperatures, or under provocation from different data, I/O or external interference conditions. A typical clue is failure to respond to power-on, requiring a manual reset.

The sure pointer to the problem is a system which always works with SWTBUG* or MIKBUG** activated, but presents intermittent problems when an EPROM monitor is switched in, even using a known-good monitor chip.

A "good-engineering-practice" pull-down for a Schottky input is 2ØØ ohms. However, other chips have trouble pulling up 2ØØ ohms when the switches are closed — so a compromise value of 47Ø to 75Ø ohms may be indicated.

*TM Southwest Technical Products Corp.
**TM Motorola, Inc.

If you're suffering from the problem described, try tacking a 1k to 3k resistor across each of the pull-downs. If this stops the problem, consider the preferred solution, which is to change IC13 back to its original design type, 74LS139. (It is no longer necessary to pay ridiculous prices or wait a year to get one of these chips.)

Leaving R6 and R8 at 1k will provide adequate pull-down and noise margin for the low-power chip, and be more compatible with the other chips which must drive these bus lines.

DMAF-1/DMAF-2 Systems Using SWTPC 68ØØ CPU

Geoffrey A. Gass
524Ø S. W. Dosch Rd.,
Portland, Oregon 972Ø1

As many frustrated SWTPC customers have learned, that firm has quietly decided to abandon entirely any further support of 68ØØ-based systems, and is deleting all 68ØØ-oriented products from its line — including the DMAF-1 single-density floppy-disk system.

It is still possible for owners of SWTPC 68ØØ CPU's to add the dual floppy disk with DMA — but it's considerably more difficult and expensive now.

SWTPC offers the DMAF-2 double-density dual floppy disk system (same drives; different controller) for a few hundred dollars more than the DMAF-1, and the hardware is nominally 68ØØ-compatible. However, SWTPC ships this only with a 68Ø9 FLEX operating system. To obtain a 68ØØ/DMAF-2-compatible operating system, one must go to the original authors of FLEX, Technical Systems Consultants (consult their ads in this journal for the address). The new DOS differs from the DMAF-1 version in the NEWDISK formatting (now compatible with the requirements of the 1791 floppy-disk controller chip), and in clearing the "double density" bit in the drive-select word (the old 68ØØ hardware is too slow to handle the double-density mode of the DMAF-2).

Some special problems may remain for those who hope to interchange disks between older DMAF-1 and newer DMAF-2 systems. The 1791 FDC chip in the DMAF-2 cannot read disks formatted by the original NEWDISK routine of the DMAF-1 FLEX 1.Ø operating system (though the 1771 in the old system has no problem reading disks formatted for the 1791!).

Therefore, all disks used for software or data interchange between DMAF-1 and DMAF-2 systems must be initialized using the DMAF-2 FLEX NEWDISK command. This formatter puts $FF's in various gaps on the disk (where the old NEWDISK put ØØ's). Once properly formatted, the disk can be written on via either system and read via either system -- no problem once the original formatting is correct.

Some users may have DISKBUG* or a similar PROM already having a disk boot in it written for the DMAF-1. This boot unfortunately fails to clear the Double-Density bit in the DMAF-2 Drive-Select Latch, and so is not directly compatible with the DMAF-2. However, since the DMAF-2 must be locked to single-density operation with the 68ØØ, it can be hard-wired as follows:

Cut the line that runs from pin 16 of IC29, the Drive Select Latch on the DMAF-2 controller board, to IC2Ø, IC15 and IC7, just after the line leaves IC29 and before it goes through the feed-through hole near R55 and D6. Install a 1Øk 1/4W pullup resistor in this feed-through hole, picking up the +5V supply at the other end of R55. This change forces single-density operation, regardless of the state of the Density bit in the Drive Select Latch.

Some software modifications to SWTBUG based assembler tools

A.J. HALL
Diagnostic Ultrasound Unit,
Queen Mother's Hospital, Glasgow G3 8SH
Scotland.

It is very useful to have a hard copy facility when developing and modifying programs and this need is recognised in the South Western Technical Products (SWTP) 6800 computing system by providing software routines to drive a PR40 parallel printer via an MP-L interface located at port 7. Typical of software expecting this configuration is Ed Smith's Software and SWTP software. However, because the data transfer is parallel and controlled by handshaking the code is not suitable for driving serial devices such as a teletype. While a 110 band teletype is slow and noisy these are not major disadvantages if it is used for listing only. Additionally they are often readily available for use because they are being superseded by video terminals in the large computer installations used in universities etc.

To use a teletype with the software mentioned above it is easiest to replace the MP-L interface with an MP-S ACIA interface and alter the printer initialisation and printer output routines to suit. If you wish to use a port other than port 7, port 0 for example it is a simple matter to alter the "LDX X'801C' to LDX X'8000' ". In modifying Ed Smith's Software it was necessary to insert a routine to generate a line feed on detecting a carriage return but only in the loader character output routine - it is not needed elsewhere. However if used and there is space to get it in, because the ACIA output routines are shorter than the MP-L ones, it gives a double spaced listing which can be useful for editing and correcting.

Ed Smith's Assembler and Loader

The M68AS assembler and relocating linking loader (version 2) is fully documented and the availability of a concise source code listing makes modification simple and straightforward.

The Loader - this provides the user with a print option to list the entry point addresses. The printer routines are found under the heading "PRINT ENTRY TABLE ON PR40' which starts on page 17 of the loader listing.

(A) PORT CONFIGURATION - The code for this starts on line 883 ($0755) at the label "PRINT IN" and continues up to but does not include "LDX PNTMSG". This code should be altered to that given below.

```
0755 CE801C       LDS #X'801C'
0758 C6 03        LDAB #X'03'
075A E7 00        STAB 0,X
075C C6 45        LDAB #X'45'
075E E7 00        STAB 0,X
0760 01 01        NOP, NOP
0762 01 01        NOP, NOP
0764 CE 07D8      LDX PNTMSG
```

(B) SINGLE COLUMN LISTING - This is obtained by changing the code on the two lines following the comment "PRINT ADDRESS" on Page 18 to provide a carriage return and branch to the "PRINT" output routine

```
0786 8D 06 (078E) BSR $ 3 PRINT ADDRESS
0788 86 0D        LDAA #X'0D'
078A 8D 32 (07BE) BSR PRINT
```

(C) CHARACTER OUTPUT - To output a character the "PRINT" routine starting at 07BE is modified to drive an ACIA and to provide a line feed ($0A) on detecting a carriage return code ($0D). The new listing is as below

```
07BE DF 82  PRINT  STX TEMP
07C0  37           PSHB
07C1 CE 801C       LDX#X'801C'
07C4 C6 02   F2    LDAB#X'02'
07C6 E5 00   WAIT  BIT B 0,X
07C8 27 FC         BEQ WAIT
07CA A7 01         STAA 1,X
07CC 81 0D         CMPA#X'0D'
07CE 26 04         BNE F1
07D0 86 0A         LDAA#X'0A'
07D2 20 F0         BRA F2
07D4 33      F1    PULB
```

The Assembler - This, like the loader, is configured for parallel output to the printer port and will not support teletype operation even though there is some provision for it in the software; the print command description (p 11) in the instructions tells the user to enter 72 for teletype use. When the printer initialisation and output routines found in the section "PRINTER INTERFACE ROUTINES" are modified, an entry of 72 allows the teletype to be used but only if the following bug on page 3 is fixed. The instruction on line 200 ($00CA) should be changed from "JSR PUT.CHR" to "JSR OUTPTR".

Port Configuration - The printer port initialisation routine is contained between linea 88 ($0000) and 96 ($000F) and starts with the label "INIT.PNT"; it should be replaced with the initialisation code given above in the section on loader modification. After the last line of the replacement code, put in NOP's down to the RTS.

Character Output - The output routine is contained between line 213 ($00E3) and line 224 ($00F9) and starts with the label "OUT.PTR"; it should be replaced with the code given below

```
00E3 CE 801C  OUT.PTR  LDX# X'801C'
00E6 C6 02             LDAB#X'02'
00E8 E5 00    WAIT     BIT  B 0'X
00EA 27 FC             BEQ WAIT
00EC 16                TAB
00ED C4 7F             ANDB#X'7F'
00EF E7 01             STAB 1,X
        Then NOP's until
00F9 39                RTS
```

Ed Smith's Disassembler and Trace (V2.5)

This package allows the user to examine, change and execute a program line by line under complete operator control when debugging a program. A print option allows all or part of a listing to be transferred to printer port 7, but again the configuration is for a PR40 printer. In casette form the program is supplied with 3 different origins $3700, $2700 and $1700. The code listing given below is for the program at $2700. As no source listing is supplied with the package the code changes given below for printer initialisation and output are shown in a disassembly listing.

Port configuration starts at $27CE with a LDX $2715 which is where the port address is stored.

```
27CE FE LDX    $2715
27D1 C6 LDAB #$ 03
27D3 E7 STAB    00,X
27D5 C6 LDAB  #$45
27D7 E7 STAB    00,X
27D9 C6 LDAB  #$3F *
27DB 01 NOP
27DC 01 NOP
27DD F7 STAB   $27E8 *
27E0 39 RTS
```

* These codes must be included.

Character Output - starts at $27B6; again with a call for the port address

```
27B6 FE LDX   $2715
27B9 C6 LDAB #$02
```

```
27BB D5 BIT B   00,X
27BD 27 BEQ   $FC  ($27BB)
27BF A7 STAA 01,X
```

Then NOP's down to

```
27C9 33 PULB
27CA FE LDX   $27E9
27CD 39 RTS.
```

SWTP CORES Assembler (V2.0)

It does not appear possible to obtain a source code listing of this product from the supplier but some information about it is to be found in the literature (1). The routines are the same as previously discussed.

Port Configuration - This starts at $17A2 with a LDX $ 801C and finishes at line $17AB; the printer initialisation code listed in the modifications to Ed Smith's loader can be used but do not include the NOP's - stop at the STAB, 0,X instruction.

Character output - the code for this starts at $1A83 with a STX $01CA to save the index register which is then loaded with the printer port address. From $1A89 onwards substitute the printer output routine given in the section on modifying the Ed Smith's assembler. After entering the necessary code put in NOP's down to $1A99 just before the RTS located at $1A9C.

Reference

(1) GASS, G.A. 1979 Patching the SWTPC Co-Resident Editor Assembler. Dr. Dobb's Journal of Computer Catisthenics and Orthodontia. Vol. 4 pp 28-36.

THE BIT BUCKET

Where all that 'good stuff' falls.
Something for everyone.

529 Fourth Avenue
Bethlehem, PA 18018
August 11, 1980

'68' Micro Journal
[illegible] Mamill Rd.
P. O. Box 849
Clinton, Tennessee 37343

Dear Sirs:

Some time ago, there was a letter in the 'Help' column requesting information on the interface to a TI-58 calculator. As I have a TI-59, I have a similar interest.

The following articles are available from:

Maurice E. T. Swinnen
Texas Instruments Software Consultant
9213 Lanham Severn Road
Lanham, Maryland 20801

'Texas Instruments PC-100A Interface Description' 12 pages

'CRT Interface for TI-59' by J.Janson and S. Boyer 25 pages

Circuit diagram of TI-59 1 page

If I ever find the time, I hope to design hardware and software for the interface, unless I buy an AR9511 or AR9512 first. There seems to be enough information in the above at least for replacing the PC-100 or with internal wiring simulating the keyboard.

Sincerely yours,

Richard E. Dell

PERCOM
PERCOM DATA COMPANY, INC.

NEWS RELEASE

RELEASE: Immediate
CONTACT: Harold Mauch
(214) 272-3421

PERCOM MANUFACTURING VIDEO DISPLAY CONTROLLER
FOR EXORCISER® BUS SYSTEMS

Garland, Texas - April 23, 1980 - Harold Mauch, president of Parcon Data Company, announced here today that the company is now manufacturing a video display generator/controller module for the EXORciser® and EXORciser®-bus compatible systems.

The VDC card, designated the VC-EX(tm) is the third member of the Porcom ModulEX(tm) family of low-cost EXORciser® bus compatible modules.

Other ModulEX(tm) units include the LFD-400EX(tm) mini-disk system and the M64EX(tm) 64-Kbyte RAM card.

Mauch said the VC-EX(tm) is memory resident and therefore the host MPU has direct access as for any other memory.

He said the memory residency feature means the display is instantly updated, and program control of display characteristics is accomplished by straightforward memory addressing and data

Display characteristics that may be software controlled include data line length, number of data lines per frame, scrolling, cursor positioning, character intensity and scanning mode (interlaced or non-interlaced).

The display may also be easily modified for reverse video, i.e., black characters on white background.

The VC-EX(tm) includes on-card RAM to accommodate all display control registers and provide for character store memory. The on-card RAM may be strapped for residency in either of two memory space locations.

The VC-EX(tm) character generator generates 128 characters, in a 7x9 dot matrix, including English letters with true lower case baseline descenders, Greek letters, numerals, standard correspondence symbols and special symbols.

Other features of the VC-EX (tm) include:

* A buffered parallel input port which allows the VC-EX(tm)

to accommodate an encoded or unencoded keyboard, a panel of membrane switches or other input device.

- Provision for an optional 2716 EPROM for use in generating up to 128 additional symbols or characters.
- Provision to interrupt MPU operation using a CRT sync or blanking signal. This feature allows the processor to update during video blanking, for example, to permit

totally "glitch-free" displays.

- Provision for "valid memory address" control by any of three system bus lines, as might be required if the bus is shared by more than one system, or if more than one VC-EX(tm) card is installed in the same system bus.

An optional VC-EX(tm) driver program, in either 6800 or 6809 code, is available in ROM and on mini-diskette. A ROM driver may be installed in a socket provided on the Percom MPU ModulEX(tm) card. The MPU operating system, PSYMON(tm), and the video driver auto-link in a way in which the driver serves to extend the PSYMON(tm) operating system, and in which PSYMON(tm) I/O is vectored to the video driver.

Orders for the VC-EX(tm) may be placed or additional literature requested by dialing Percom's toll-free order number, 1-800-527-1592. OEM quantity pricing schedule available on request.

(tm) trademark of Percom Data Company, Incorporated.

* trademark of Motorola Corporation.

We first became interested in T.S.C.'s software products with their introduction of BASIC and SORT/MERGE for the 6809. As users of a 6809 EXORciser, our first task was to convert FLEX 9.0 to run on the EXORdisk. In doing so, it was discovered that several desirable utilities were either not available or were not compatible with our disk driver routines. As a result, we find ourselves with a growing library of utilities and other programs which we would like to share.

This month, we are submitting a BACKUP utility. It will run on any FLEX 9.0 system since it makes no references to any function outside of FLEX itself. BACKUP copies all sectors from the diskette in drive 00 to the diskette in drive 01. The result is an exact duplication of the diskette in drive 00. BACKUP executes in 2.5 minutes, compared to as much as 10 minutes for the FLEX COPY command copying a full disk. However, BACKUP will not reorganize the files copied to drive 01.

To use BACKUP, simply enter "BACKUP". The user is then asked if he wishes to backup from drive 00 to 01. If a "Y" is entered, the user is asked "ARE YOU SURE?". If another "Y" is entered, BACKUP proceeds to copy drive 00 to drive 01. In either case, FLEX is entered if a response other than "Y" is received. A data diskette may be backed by exchanging the diskette in drive 00 before typing a "Y" response. All data on drive 01 will be overwritten with data from drive 00. Flex is entered upon completion of BACKUP. It is not necessary to link the DOS since all sectors are an exact copy of the original.

BACKUP staggers sector reads and writes so that minimum time is wasted due to disk rotational delays. Reading and writing every third physical sector provides the fastest possible I/O with the disk drivers we are using. If your system does not execute BACKUP in approximately 2.5 minutes, then you will have to change the sector geometry at label "[illegible]". The sectors on each track are physically arranged in the following order: 1,6,11,3,8,13,5,10,15,7,12,2,9,14,4.

With minor changes, BACKUP should also run for the minidisk version of 6809 FLEX. We suggest the following changes: At lines 47 and 51 change the "15" in the expression to the number of sectors per track for the minifloppy; At lines 120 and 161 change the "77" to the number of tracks per diskette.

```
00001                                ******************************************************
00002                                *                                                    *
00003                                *              FLEX 9.0 BACKUP UTILITY               *
00004                                *                                                    *
00005                                *                    WRITTEN BY                      *
00006                                *                 RICHARD F. SCOTT                   *
00007                                *                                                    *
00008                                *               COPYRIGHT (C) 1980 BY                *
00009                                *            CAROLINA COMPUTER PRODUCTS              *
00010                                *              438 WEST [illegible] BOULEVARD        *
00011                                *             WINSTON-SALEM, N.C.  27101             *
00012                                *                                                    *
00013                                * A HIGH SPEED DISK-TO-DISK BACKUP UTILITY DESIGNED  *
00014                                * TO BE OPTIONALLY CALLED FROM A USER PROGRAM. SECTOR *
00015                                * READS AND WRITES ARE STAGGERED SUCH THAT MINIMUM   *
00016                                * TIME IS WASTED DUE TO DISK ROTATIONAL DELAYS.      *
00017                                * THIS UTILITY USES ALL AVAILABLE MEMORY SPACE AS A  *
00018                                * DISK DATA BUFFER.  ALL SECTORS ARE COPIED.         *
00019                                *                                                    *
00020                                ******************************************************

00022                                ** DEFINE FLEX 9.0 ENTRY POINTS AND VARIABLES **

00024               CD18     A PUTCHR EQU    $CD18     OUTPUT CHAR TO CONSOLE.
00025               CD15     A GETCHR EQU    $CD15     INPUT CHAR FROM CONSOLE.
00026               CD3F     A RPTERR EQU    $CD3F     REPORT ERROR.
00027               DE00     A READ   EQU    $DE00     READ A SECTOR.
00028               DE03     A WRITE  EQU    $DE03     WRITE A SECTOR.
00029               DE09     A RESTOR EQU    $DE09     RESTORE HEAD TO TRACK 0.
00030               DE0C     A DRVSEL EQU    $DE0C     SELECT DRIVE.
00031               CD03     A DOS    EQU    $CD03     FLEX ENTRY.
00032               C840     A DOSFCB EQU    $C840     FLEX'S UTILITY FCB AREA.
00033               CC2B     A MEMEND EQU    $CC2B     ADDR OF MEMORY END.

00035A C100                           ORG    $C100     RELOCATABLE.
00036A C100 20      09   C10B BACKUP  BRA    START     SKIP VARIABLES.
00037A C102         01     A VN       FCB    1         VERSION.
00038A C103         00C0   A          FDB    BKUP-*    OFFSET TO BACKUP SUBR.
00039A C105         00     A TRK      FCB    0         TRACK COUNTER.
00040A C106         00     A NUMTRK   FCB    0         BLOCK SIZE, IN TRACKS.
00041A C107         0000   A BUFADR   FDB    0         BUFFER START ADDR.
00042A C109         0000   A SPSAV    FDB    0         STACK SAVE AREA.

00044A C10B FC      CC2B   A START    LDD    MEMEND    GET MEMORY END ADDR.
00045A C10E 6F      8C F5             CLR    NUMTRK,PCR NUMTRK WILL HOLD TRACKS TO COPY
00046A C111 C3      0001   A          ADDD   #1        SET TO 1 BEYOND MEMORY END.
00047A C114 83      0F00   A SUBTRK   SUBD   #(256*15) SUBTRACT TRACK SIZE.
00048A C117 25      05   C11E         BCS    CHKTRK    CONT. IF HAVE BORROW.
00049A C119 6C      8C EA             INC    NUMTRK,PCR BUMP TRACK COUNT.
00050A C11C 20      F6   C114         BRA    SUBTRK
00051A C11E C3      0F00   A CHKTRK   ADDD   #(256*15) RESTORE TO BEFORE UNDERFLOW.
00052A C121 1F      01     A          TFR    D,X
00053A C123 6D      8C E0             TST    NUMTRK,PCR
00054A C126 26      40   C168         BNE    QUERY     OK IF ROOM FOR AT LEAST 1 TRK.
00055A C128 17      0094 C1BF         LBSR   PSTR      PRINT FOLLOWING ERROR MSG.
00056A C12B         2A     A          FCC    /** INSUFFICIENT MEMORY - AT LEAST /
00057A C14[illegible] 31   A          FCC    /1024 BYTES REQUIRED **/
00058A C163         0700   A          FDB    $0700
00059A C165 7E      CD03   A EXIT     JMP    DOS       RE-ENTER FLEX.

00061A C168 8D      55   C1BF QUERY   BSR    PSTR
00062A C16A         0D0A   A          FDB    $0D0A
00063A C16C         42     A          FCC    "BACKUP FROM DRIVE 0 TO 1 (Y/N)? "
00064A C18[illegible] 00   A          FCB    0
00065A C18D BD      CD15   A          JSR    GETCHR    GET RESPONSE.
00066A C190 81      59     A          CMPA   #'Y       IS IT "Y"?
00067A C192 1C      FE                CLC
00068A C194 26      CF   C165         BNE    EXIT      EXIT IF NOT.
00069A C196 8D      27   C1BF         BSR    PSTR
00070A C198         0D0A   A          FDB    $0D0A
00071A C19A         41     A          FCC    /ARE YOU SURE? /
00072A C1A8         00     A          FCB    0
00073A C1A[illegible] BD CD15 A       JSR    GETCHR
00074A C1A[illegible] 81 59   A       CMPA   #'Y       RESPONSE IS "Y"?
00075A C1A[illegible] 1C FE           CLC
00076A C1B0 26      B3   C165         BNE    EXIT      EXIT IF NOT.
00077A C1B2 8D      0B   C1BF         BSR    PSTR
00078A C1B4         0D     A          FCB    $0D,$0A,0
00079A C1B7 A6      8C F[illegible]   LDA    NUMTRK,PCR SET UP FOR BACKUP FUNCTION.
00080A C1BA 8D      11   C1CD         BSR    BKUP
00081A C1BC 20      A6   C165         BRA    EXIT

00083                                * PRINT STRING AT CALLER'S RETURN ADDR.

00085A C1BF 34      16     A PSTR     PSHS   D,X
00086A C1C1 AE      64     A          LDX    4,S       X->STRING.
00087A C1C3 A6      80     A PSTR2    LDA    ,X+
00088A C1C5 27      05   C1CC         BEQ    PSTR1     EXIT IF NULL CHAR.
00089A C1C7 BD      CD18   A          JSR    PUTCHR    OUTPUT CHAR.
00090A C1CA 20      F7   C1C3         BRA    PSTR2
00091A C1CC AF      64     A PSTR1    STX    4,S       SET RETURN ADDR.
00092A C1CE 35      96     A          PULS   D,X,PC    RESTORE D,X; RETURN.

00094                                * ACTUAL BACKUP FUNCTION IS DESIGNED AS A SUBROUTINE
00095                                * SO THAT OTHER PROGRAMS CAN HAVE ACCESS.
00096                                * ENTRY REQUIREMENTS: X->BACKUP BUFFER START ADDRESS,
00097                                * ACCA CONTAINS NUMBER OF TRACKS TO COPY PER LOOP
00098                                * (REQUIRED BUFFER SIZE IS 256*15*ACCA). EXIT CONDITIONS
00099                                * NO REGISTERS ALTERED EXCEPT CC REGISTER, WHERE C=0
00100                                * IF NO ERRORS OCCURRED AND C=1 IF ERROR OCCURRED. IF
00101                                * ERROR, MESSAGE IS DISPLAYED BEFORE RETURN.
```

```
C100 34    7E        A BKUP    PSHS  DP,D,X,Y,U SAVE ALL.
C102 10EF  8D FF32             STS   SPSAV,PCR SAVE ENTRY STK.
C107 A7    8D FF38             STA   NUMTRK,PCR SAVE NUMBER TRACKS PER COPY.
C10B AF    8D FF3B             STX   BUFADR,PCR SAVE BUFFER START ADDR.
C10F 8D    30    C141          BSR   SEEK0    RESTORE HEADS TO TRK 0.
C111 6F    8D FF2D             CLR   TRK,PCR  ZERO TRACK COUNTER.
C115 8D    3A    C151 BKUP2    BSR   TGLDRV   SELECT DRIVE 0.
C117 108E  DE00      A         LDY   #READ    SET UP FOR READ.
C11B 8D    42    C15F          BSR   RDWRT    READ (NUMTRK) TRACKS.
C11D E7    8D FF14             STB   TRK,PCR  RESTORE BEGINNING TRACK #.
C121 8D    1E    C141          BSR   TGLDRV   SELECT DRIVE 1.

C123 108E  DE03      A         LDY   #WRITE   SET UP FOR WRITE.
C127 8D    36    C15F          BSR   RDWRT    WRITE (NUMTRK) TRACKS.
C129 E7    8D FF0B             STB   TRK,PCR  RESTORE BEG TRACK #.
C12D 108E  DE00      A         LDY   #READ    SET UP FOR READ.
C131 8D    2C    C15F          BSR   RDWRT    READ TO VERIFY CRC.
C133 A6    8D FEF2             LDA   TRK,PCR
C137 81    4D        A         CMPA  #77      COPIED ENTIRE DISK?
C139 25    DA    C115          BLO   BKUP2    DO MORE IF NOT.
C13B 8D    11    C14E          BSR   SEEK0    RESTORE HEADS TO TRK 0.
C13D 1C    FE                  CLC            SET NO-ERROR CONDITION.
C13F 35    FE        A         PULS  DP,D,X,Y,U,PC RETURN.

               * SWITCH SELECTED DRIVE FROM 0 TO 1 OR 1 TO 0.

C141 8E    C840      A TGLDRV  LDX   #DOSFCB
C144 A6    03        A         LDA   3,X      GET DRIVE SPECIFIED.
C146 43                        COMA           COMPLIMENT AND MASK UNUSED.
C147 84    01        A         ANDA  #1
C149 A7    03        A         STA   3,X
C14B 7E    DE0C      A         JMP   DRVSEL   SELECT DRIVE & RETURN.

               * RESTORE HEADS TO TRACK 0, DRIVE 0 AND 1.

C14E 8D    F1    C141 SEEK0    BSR   TGLDRV   X-->DOSFCB.
C150 6F    03        A         CLR   3,X      SELECT DRIVE 0.
C152 BD    DE09      A         JSR   RESTOR   SEEK TO TRK 0.
C155 26    80    C267          BNE   DKERR    DO IF ERROR.
C157 8D    E8    C141          BSR   TGLDRV   SELECT DRIVE 1.
C159 BD    DE09      A         JSR   RESTOR   SEEK TO TRK 0.
C15C 26    39    C267          BNE   DKERR    DO IF ERROR.
C15E 39                        RTS

               * READ OR WRITE SPECIFIED NUMBER OF TRACKS, NOT
               * EXCEEDING 77TH TRACK. NUMBER OF TRACKS TO READ/WRITE
               * IN NUMTRK, STARTING TRACK NUMBER IN TRK, BUFFER
               * START ADDRESS IN BUFADR. ON EXIT, TRK UPDATED,
               * ACCB CONTAINS ORIGINAL STARTING TRACK NUMBER.

C15F AE    8D FED4    RDWRT    LDX   BUFADR,PCR X-->BEG OF BUFFER.
C163 A6    8D FECF             LDA   NUMTRK,PCR NUMBER OF TRACKS TO COPY.
C167 E6    8D FECA             LDB   TRK,PCR  CURRENT TRACK.
C16B 34    06        A         PSHS  A,B      SAVE.
C16D 31    8C 1D      RDWRT2   LEAY  <SCTORD,PCR STAGGER SECTORS FOR MAX SPEED.
C170 A6    8D FEC1    RDWRT3   LDA   TRK,PCR  GET TRACK NUMBER.
C174 E6    C0        A         LDB   0,U+     GET SECTOR NUMBER.
C176 2B    0A    C182          BMI   RDWRT4   DONE WITH TRACK IF NEG.
C178 AD    A4        A         JSR   0,Y      DO READ OR WRITE.
C17A 8D    18    C194          BSR   CHKERR   CHECK FOR ERRORS.
C17C 30    89 0100 A           LEAX  256,X    BUMP BUFFER PTR.
C180 20    EE    C170          BRA   RDWRT3   GO UNTIL END OF TRACK.
C182 6C    8D FEAF    RDWRT4   INC   TRK,PCR  BUMP TRACK NUMBER.
C186 A6    8D FEAB             LDA   TRK,PCR
C18A 81    4D        A         CMPA  #77      AT END OF DISK?
C18C 24    04    C192          BHS   RDWRT5   TO RDWRT5 IF SO.
C18E 6A    E4        A         DEC   0,S      READ/WRITTEN (NUMTRK)?
C190 26    DB    C16D          BNE   RDWRT2   DO ANOTHER TRACK IF NOT.
C192 35    86        A RDWRT5  PULS  D,PC     (ORIGINAL TRK IN ACCB.)

               * CHECK FOR DISK ERROR. IF NO ERROR, RETURN. IF
               * ERROR, CONVERT ERROR STATUS BYTE (IN ACCB) RETURNED
               * BY DISK DRIVERS TO ERROR CODE ACCEPTABLE TO FLEX,
               * AND DISPLAY STANDARD FLEX ERROR MESSAGE.

C194 26    01    C197 CHKERR   BNE   DKERR    ERROR IF .NE. ZERO.
C196 39                        RTS
C197 31    8C 1D      DKERR    LEAY  <ERRCO,PCR ERROR CONVERSION CODES.
C19A 86    06        A         LDA   #6       DO FOR 6 LOOPS MAX.
C19C E1    A1        A DKERR2  CMPB  0,Y++    MATCHING ERROR CODE?
C19E 27    07    C1A7          BEQ   DKERR3   GO IF MATCH.
C1A0 4A                        DECA
C1A1 26    F9    C19C          BNE   DKERR2   LOOP 6 MAX.
C1A3 C6    FF        A         LDA   #255     IF NOT FOUND, USE ERROR $255.
C1A5 20    02    C1A9          BRA   DKERR4
C1A7 A6    3F        A DKERR3  LDA   -1,Y     GET CONVERTED ERROR CODE.
C1A9 A7    01        A DKERR4  STA   1,X      SET ERROR CODE IN FCB.
C1AB BD    CD3F      A         JSR   RPTERR   LET FLEX ANNOUNCE ERROR.
C1AE 1A    01                  ORCC           SET ERROR CONDITION.
C1B0 10EE  8D FE84             LDS   SPSAV,PCR RESTORE STACK PTR.
C1B5 35    FE        A         PULS  DP,D,X,Y,U,PC RETURN WITH ERROR COND.

               * THE FOLLOWING TABLE CONVERTS ERROR CODES AS
               * GENERATED BY THE DISK DRIVERS TO AN ERROR
               * CODE ACCEPTABLE TO FLEX'S 'RPTERR' ROUTINE.

C1B7       08        A ERRCO   FCB   $08,9    TRANSLATE "8" TO "9",
C1B9       40        A         FCB   $40,11   "$40" TO 11,
C1BB       80        A         FCB   $80,16   ETC., ETC.
C1BD       20        A         FCB   $20,8
C1BF       10        A         FCB   $10,14
C1C1       18        A         FCB   $18,9

               * SECTORS READ/WRITTEN FOR EVERY TRACK ARE
               * STAGGERED FOR FASTEST POSSIBLE I/O.
               * PATTERN IS EVERY 3RD PHYSICAL SECTOR.

C1C3       01        A SCTORD  FCB   1,3,5,7,9,6,8,10,12,14,11,13,15,2,4
C1D2       FF        A         FCB   -1       NEG VALUE IS TERMINATOR.
C1D3       0020      A         RMB   32

           C100      A         END   BACKUP
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000


BACKUP C100  BKUP   C100  BKUP2  C115  BUFADR C107  CHKERR C194  CRTTRK C118
DKERR  C267  DKERR2 C30C  DKERR3 C277  DKERR4 C279  DOS    CD03  DOSFCB C840
DRVSEL DE0C  ERRCO  C287  EXIT   C165  GETCHR CD15  INBUFF CC28  NUMTRK C106
RSTP   C1BF  PSTR3  C1C1  PSTR3  C1CC  PUTCHR CD18  QUERY  C184  RDWRT  C22F
RDWRT2 C23D  RDWRT3 C240  RDWRT4 C252  RDWRT5 C262  READ   DE00  RESTOR DE09
RPTERR CD3F  SCTORD C293  SEEK0  C21E  SPSAV  C109  START  C100  SYSTEM C114
TGLDRV C211  TRK    C105  VN     C102  WRITE  DE03
```

Don Williams Sr.
'68' Micro Journal
POB 849
3018 Hamill Road
Hixson, Tennessee 37343

21 April 1980

Dear Don:

We would deeply appreciate your announcing the formation of a 68XX user's group in the San Diego area. Anyone interested should contact:

Mel Zeddies
1854 Pacific Beach Drive
San Diego, CA 92109
(714) 274-4087

Many thanks.

Keep up the good work!

Sincerely,

Mel Zeddies

7768
Benutzergruppe

Peter Bendall
Flottmoorring 67
D-2358 Kaltenkirchen
West Germany
Tel 04191 6538

Dear Mr Williams

I am one of many satisfied users of a non-6850 syst designed and sold in Europe by Newbear Computing Store of Newbury in England. It was developed to provide a home-brew system for people who hadnt time or maybe ability to develop one for themselves. In my experience it has succeeded by doing everything simply and reasonably cheaply, no trick circuits for instance! Its two disadvantages of course, no 6850, and 625kHz Clock. My System has 36K of RAM, Mikbug compatible Monitor using a memory mapped VDU, and runs TSC software under CFM. TSC BASIC certainly makes up for the slow speed. The system doesn't stop at just all the functions you would need, even a 6809 CPU card and a floppy controller are available in kit form.

There are just a couple of things that someone over there might be able to do for me and my friends. Like maybe someone is running a SS3 Cassette Interface at 4800 Baud, and would like to swap a couple of tapes so I can set up my CFM loops to read and write? Since the 7768 from Newbear has 625 kHz instead of a round 1000 it needs to get smaller loops. The second little thing Id like to find is a Floppy Operating system for 5 inch Hard sectored floppies that I can Customise, or any ideas how I can read soft sector floppies with a hard sector controller.

We look forward to hearing from you over there and were always pleased to swap letters, ideas and software with anyone.

Best wishes, and thanks for the magazine

Peter Bendall
(DJ0JR,G3NBU)

'68' Micro Journal
POB 849
Hixson, TN 37343

Dear Mr. Williams,

This letter is in regards to DYNASOFT PASCAL RELEASE 1.0. I have received a letter from Mr. Jost of Dynasoft Pascal, and he explained the delay in answering my letter to my satisfaction. He also offered an apology for the delay. Mr. Jost has been working on updates to Pascal and should have some future "goodies" for us.

Mr. Jost was kind enough to send me the information that I needed to adapt Pascal to print on PORT 03, as well as further information on using the Pascal routines. I have also talked to Mr. Jost by telephone, and I fully expect Dynasoft to back their software releases.

This Pascal may not be a full sized disk Pascal, but I feel that it is worth the price. It is also a very good learning Pascal for those might wish to move up to a full Pascal at a future date.

Thank you for the opportunity to set the record straight!

Sincerely yours,

Jerry P. Starzinski
POB 9456
Yakima, WA 98909

Memo

(803) 625-1075

To: Don Williams, publisher
68 Micro Journal
3018 Hamill Rd., Box 849
Hixson, Tennessee 37343

STEPHEN L. CARTER
ATTORNEY AT LAW
227 Railroad Ave.

REGARDING:

DATE: 25 July 80

I just made a discovery that I'd like to share with other 6800 users. I use FLEX 2.0 with Pangoo/Pertec drives which are capable of operating on 40 tracks, and I've put the patches to NEWDISK as shown in recent issues of the 68 Micro Journal. The patches work fine, except when I tried to use all forty tracks of BASF "Flexy Disk"s. When I tried to read or write to anything in tracks 36-40, I got all sorts of dreaded DISK FILE READ ERRORs. I carefully pulled out my hair, suspected awful things about my drives, and about panicked when I looked at the BASF disks, and discovered that the hole is about 1/8" shorter than on the VErbatim diskettes I've been using. The wear marks also were disturbingly close to the end of the hole. Conclusion: BASF disks won't work with 40 tracks unless you want to mutilate the data hole.

So far, my conclusion has proven true, although we lost some important data during before we discovered it.

Steve Carter

'68' Micro Journal
3018 Hamill Rd.,
PO Box 849
Hixson Tennessee
37343 U.S.A.

FLEX09 V2.6 PATCH TO USE WITH OLD SWTP BOX

Dear Editor;-

For people who use FLEX 09 Version 2.6 with old SWTP box (MPB-type mother board), the following patch may be useful.

When power up or reset, the FLEX initializes I/O ports by 16-address-space step, thus everytime you restart you have to relocate I/O addresses by 4-space step using SBOX.CMD unless you have newest SWTP models such as S/09 or 69A. This patch allows to initialize by 4-space step as old version of FLEX did.

SBOX.CMD still works after this modification.

Address	Original Data	Patched
$CC33	00	04

Yours truly,

K.Mitadera
126 Sedgefield,
Pointe Claire,
Quebec CANADA
H9R 1N5

Peter Murray
P.O. Box 49302
Austin, TX 78765
July 21, 1980

Don Williams, Editor
'68' Micro Journal
3018 Hamill Road
Hixson, TN 37343

Dear Don,

I would like to take this opportunity to inform your readers that JCP is now being supplied with an additional program that will allow procedure calls from a mainline procedure.

LIBJCP is used as a FLEX* command within a procedure to load and execute another procedure. Upon conclusion of the called procedure, control will return to the mainline procedure and execution will resume at the line following the LIBJCP command.

With the addition of LIBJCP, the user now has another level of control at his (or her) disposal. For example, a procedure can now be coded to control the execution of several procedures.

Sincerely,

Peter Murray

* FLEX is a trademark of Technical Systems Consultants, Inc.

S D O S T I D B I T S

"SDOS will soon announce a microcomputer network (SDNET). SDNET is a shared resource microcomputer-based network for locally distributed data processing and data management. See the first sub-group newsletter for details.

"SDOS is now available for EXORciser users, along with the SD BASIC Compiler, Assembler and Editor. The EXORciser SDOS also supports a 10 megabyte Winchester hard disk (available from SD).

"A document processing program, TYPE, has been released by SD. TYPE is suitable for generating letters or large documents. TYPE allows complete control of line widths, page depth, margins, page numbering, titles, and footnotes. Right and left margin justification is performed automatically. A merge option allows generation of form letters with customized names and/or data inserted anywhere. Table of Contents generation and Dewey decimal notation are also supported.

"SD is currently working on a business package, including Accounts Payable, Accounts Receivable, and General Ledger.

"The long awaited BASIC V1.4 Compiler should be available to end-users September 1st, 1980. This compiler provides line labels, parameterized multi-line functions and subroutines, string arrays, and smaller/faster compiled code.

"SDOS/MT, a multi-terminal version of SDOS is now available (initially on Midwest Scientific Instruments equipment) which handles four users. SDOS/MT supports hard disk, floppies, and line printers, in a totally device independent fashion. A special Virtual Terminal driver makes application programs independent of CRTs by defining a standard SDOS I/O call to perform CRT cursor positioning, etc. BASIC V1.4 has language constructs to make this facility easy to use.

"6809 versions of SD's popular software will also soon be available."

C O M P U T E R W A R E

PROUDLY ANNOUNCES

CSS RANDOM BASIC VER 9.0 / VER 2.0 FOR SSB DISK USERS

Computerware's tribute to the 68XX family! our newest 6800 and 6809 RANDOM BASIC's!! As one of the major users of RANDOM BASIC for software development, we know what features a powerful BASIC should have. We have exceeded even our own goals and again provided the 68XX community with the most powerful Basic Interpreter available. Version 9.0 (6800) and 2.0 (6809) are not just the product of some systems house.... they are the culmination of over 3 years of joint effort between systems and applications programmers. Ease of program development, debugging, and documentation are the type of factors our new versions concentrated on, as seen in the overview below.

VERSION 9.0 / 2.0 OVERVIEW OF NEW FEATURES

- 126 character variable labels. BASIC uses the first 6 to establish uniqueness, but the variable names may be up to 126 characters.
- IF ... THEN ... ELSE ... to allow greater ease of program flow and tighter, faster program execution.
- AUTO program line numbering allows faster, cleaner program entry. You choose the starting line number and the increment value.
- PACK and UNPACK string variables allow concatination and separation of string variables at lightning speed and with only one command (as opposed to using MID$, LEFT$, RIGHT$).
- The DO command executes DOS utility commands from BASIC.
- With extended variable names, you need the ability to CONTINUE the Disk I/O statements onto additional lines. GET, PUT, READ, WRITE can now be continued on as many lines as you like.
- Everyone has problems paging correctly on their reports. The new PAGE command was added to do the bulk of the work for you.
- MSIZE and FSIZE - were added to allow the programmer to know how much free memory space is available or how big a file is.
- BASIC automatically establishes if the system drive is single or double density and allocates file control block space accordingly. These are compatible with any version of SSB DOS from V5.0 to the new V6.0 (6800) and V1.0 thru V2.0 (6809).
- Automatic Load & Do. When you call BASIC from the disk, you may add the name of a program to be loaded and executed.
- Despite the fact that CSS RANDOM BASIC now has 110 commands, functions and statements, it is still UNDER 13K ($0010 --> $336B) in size - including all its work areas.
- What more can we say? We've concentrated on usability, flexibility, and performance, again raising the 68XX software standard. You need not accept anything less than the BEST!!

DEAR DON,

What we need in The Journal is a "beginner's" section; not just for hackers but for 'users' like myself -- little guys who still work in Basic because we can make Basic work, who feel a great thrill if we punch in a published routine and it actually runs, and who feel as if they have conquered Mt. Everest if they actually combine programs and routines (such as the attached) and make it all go.

A great amount of attention is being paid to the exciting and new; the 6809 and the 68000, to Flex 2.0 and UniFlex, to "Super Basic Version 99.9", and to a proliferation of "new" languages. Well and good. We need the news, but it should be news, not gospel. The electronic media has (at last) learned that news should be brief and concise. "Give the news, shut up, and move on to the substance of the day."

The "substance of the day" (for me, at least) in computing is still miniFlex, SWTPCo Basic if I need mathematical accuracy to some longer length, TSC Basic if I need speed, and TSC's Text Editor and Text Processor for an extremely powerful word processing system. My two little 5 inch disks still hold more data than I ever work with at one time; I have a couple of cassette machines on an AC-30 that just about never get used except to exchange programs with friends who still think like I do, and my slow poke Anderson Jacobson printer still turns out copy as fast as I can use it. My Extel Printer, at 110 baud, makes nice reference copies for me.

I envy the people with printers running at 250,000 baud; video terminals that print from top, side, and bottom at the same time (I sure like my Soroc IQ-120 at 1200 baud); and 64 bit processing systems that can output the Encyclopedia Britannica, completely formatted, in 27 seconds.

However, somehow it all seems irrelevant. Since I don't keep books for the Comptroller of the State of Texas, nor do I do research for the United States Coast and Geodetic Survey crews, I am not terribly interested in Reverse Polish Pascal running in 3 megabytes of static RAM. I am tremendously interested in a program, running in Basic, that would take the co-ordinates of Hurricane Allen (which just went over my home and that of James Caldwell), compare them to previous hurricane co-ordinates and issue a "probability plot" on future movements.

As usual, Don, I am unable to communicate in exact terms. I don't want "our" Journal following the pattern of Byte Magazine. (I have just let my subscription to Byte expire and don't intend to renew it.) DO report the new but DON'T assume we all have the resources to dash out and buy all the new goodies -- not only don't assume that we don't have the resources but please don't make the automatic assumption we have the desire to buy it. We DO have interest in it, yes.

Therefore, may we have a few more basic Basic articles, a few more miniflex programs (or better, programs written for Flex 2.0 with the corresponding miniFlex addresses given so we can do our own assemblies), and similar level.

Remember me? I'm the guy who is still trying to find some way of financing the purchase of a Gimix Mainframe to hold what cards I own. I'm sure not going to buy a new operating system until I have that. I don't feel that I stand alone in this thinking.

Best Wishes,

John

John Tucker

wkts radio 950

July 2, 1980

68 Micro Journal
P.O. 849
3018 Hamill Road
Hixson, TN
37343

Dear Sirs:

Thanks for a great publication for all of us 6[illegible]XX lovers out here. It seems that all the other consumer computer mags have decided to go with the majority and are totally ignoring all the good hardware and softw re being put toget er for the 6800 series chips.

I would be interested in hearing from anyone who has done some work on radio station TRAFFIC software. Specifically programs to h ndle logging.

We are currently doing all of our payroll, accounts receiv ble and cash flow with a dual 8 inch system and SWTPC 6800 machine with COMPUTERWARE RANDOM BASIC. Thanks to Paul Searby and his fine staff our system has paid for itself many times over during the past 2 1/2 years. Anyone not familiar with COMPUTERWARE certainly should take time to make contact.

Keep up the good work.

Yours truly,

Julian E. Jetzer
President

P.O.Box 1045 • 814 Plaza 8 • Sheboygan, Wisconsin 53081 • (414) 457-5561

Having purchased a Summagraphics bitpad for the purpose of entering EKG data into my 6809 system my first task was to write the interfacing software. This article will deal specifically with the interfacing of the bitpad to TSC'S BASIC.

First, the bitpad as purchased came with a parallel interface and was configured for a parallel binary transmission format at 200 bytes/second. The data format consistes of a continiously transmitted 5 byte sequence as follows.

BIT	7	6	5	4	3	2	1	0
BYTE1	1	BA	F3	F2	F1	F0	0	0
2	0	"	X5	X4	X3	X2	X1	X0
3	0	"	X11	X10	X9	X8	X7	X6
4	0	"	Y5	Y4	Y3	Y2	Y1	Y0
5	0	"	Y11	Y10	Y9	Y8	Y7	Y6

F0=Z AXIS SWITCH
F1, F2, F3=FLAG DATA
X0-X11= 12 BIT X COORDINATE (0 TO 2200)
Y0-Y11= 12 BIT Y COORDINATE (0 TO 2200)
BA=BYTE AVAILIABLE

The eight data lines from the bitpad are connected to the eight inputs of a PIA located at address $E01E. Also the bitpad produces a data strobe which is connected to the CB1 input of the PIA.

The first part of the interfacing is accomplished by BITPAD.BIN which when called reads in the next five byte sequence of data and places this data in a table. This program is entered at 'GETBIT' and returns with flag data at TBL, x position at TBL+1 and TBL+2 and, y position at TBL+3 and TBL+4.

The final section of the interfacing is the BASIC program BITBAS.TXT which contains two subprograms BPINIT and BITREAD. BPINIT does the following. One, places the machine program BITPAD.BIN, contained in the data statements, in the array named PROG( ). Two, BPINIT sets the address of the user function such that when USER(X) is invoked it calls BITPAD.BIN. Subprogram BITREAD when called returns with BPFLAG containing the flag data, BPXPOS containing the x position of the pen and, BPYPOS containing the y position of the pen.

The use of this program is quite simple and I hope the users of '68' MICRO JOURNAL find this a useful program. Particulary since the bitpad has proved to be quite useful and reliable.

James Taaffe
Boston University
Dept. of Physics
111 Cummington St.
Boston Ma.

```
*********
* BITPAD *
*********

* WRITTEN BY JAMES TAAFFE
*
* INPUTS 5 BYTES FROM BITPAD AND THEN PLACES THE
* DATA INTO A TABLE. UPON EXIT X POINTS TO THE START OF
* THIS TABLE.
* PIC CODE

      E01E  PIADAT  EQU    $E01E     PIA DATA REGISTER
      E01F  PIACTL  EQU    $E01F     PIA CONTROL REGISTER

                * CONFIG PIA FOR 8 INPUTS
                * TRIPS ON POSISTIVE TRANSISTION ON CB1
0000 86  3A       GETBIT  LDA    #$3A      OPEN DDR
0002 B7  E01F             STA    PIACTL
0005 7F  E01E             CLR    PIADAT    SET 8 INPUTS
0008 86  3E               LDA    #$3E      SET TRANSISTION,CLOSE DDR
000A B7  E01F             STA    PIACTL

                *NOW GET DATA
000D 30  8D 00 0  LOOP1   LEAX   TBL,PCR   LET X POINT TO TABLE
0011 8D  11               BSR    GETBYT    PUT BYTE INTO TABLE
0013 85  80               BITA   #$80      SEE IF BYTE 1 OF 5
0015 27  F6               BEQ    LOOP1     LOOP TILL HAVE IT
                * NOW GET NEXT 4 BYTS
0017 8D  0B               BSR    GETBYT
0019 8D  09               BSR    GETBYT
001B 8D  07               BSR    GETBYT
001D 8D  05               BSR    GETBYT
001F 30  8D 000E          LEAX   TBL,PCR   POINT TO TABLE
0023 39                   RTS
                *
                * GETS 1 BYTE FROM BITPAD VIA PIA
0024 B6  E01F     GETBYT  LDA    PIACTL
0027 85  80               BITA   #$80      BYTE RECIEVED
0029 27  F9               BEQ    GETBYT    LOOP TILL HAVE IT
002B B6  E01E             LDA    PIADAT    GET DAT
002E A7  80               STA    0,X+      PUT INTO TABLE
0030 39                   RTS
                *
0031              TBL     RMB    5
                          END
```

```
 *          BITPAD
 *
 *          WRITTEN BY JAMES TAAFFE
 *
 *          READS PEN POSITION AND STATUS FROM
 *          A SUMMAGRAPHIC BIT PAD
 *
 *          CALLING SEQUENCE   GOSUB BITREAD
 *          RETURNS BPFLAG BITPAD FLAG DATA
 *                  BPXPOS X POSITION OF PEN (0-2200)
 *                  BPYPOS Y POSITION OF PEN (0,2200)
 *
 *          UPON FIRST USE GOSUB BPINIT WHICH LOADS MACHINE
 *          LEVEL PROGRAM INTO ARRAY PROG( ).
 *
            DIM PROG(14)
 *          FOLLOWING DATA STATMENTS CONTAIN BITPAD.BIN
            DATA B6,3A,B7,E0,1F,7F,E0,1E,B6,3E,B7,E0,1F,30,8D,00
            DATA 20,8D,11,85,80,27,F6,8D,0B,8D,09,8D,07,8D,05,30
            DATA 8D,00,0E,39,B6,E0,1F,85,80,27,F9,B6,E0,1E,A7,80
            DATA 39,00,00,00,00,00
 *
 *
 *          LOAD PROGRAM BITPAD.BIN FROM DATA STATMENTS INTO ARRAY.
BPINIT      RESTORE:\
            P=PTR( PROG(0) ):\
            FOR I=0 TO 53:\
            READ A$:\
            POKE P+I,HEX(A$):\
            NEXT I
 *
 *          NOW SET ADDRESS OF USER FUNCTION TO POINT
 *          TO BITPAD.BIN NOW IN ARRAY PROG( ).
 *
 *          GET MEMEND POINTER
            M=PEEK( HEX('CC2B') )*256+PEEK(HEX('BC2C') )
 *          NOW FIX USER ADDRESS
            POKE M-2,(P AND 65280)/256:\
            POKE M-1,(P AND 255):\
            RETURN
 *
 *
 *          BITPAD READ ROUTINE
BITREAD     A=USR(0):\
            BPFLAG=PEEK(P+49):\
            BPXPOS=(PEEK(P+50) AND 63)*(PEEK(P+51) AND 63)*64:\
            BPYPOS=(PEEK(P+52) AND 63)*(PEEK(P+53) AND 63)*64
            RETURN
 *
            END

TOTAL ERRORS = 0
```

Painless Use of the USR Function in TSC 6809 EXTENDED BASIC

By Bud Pass

The use of the USR function in TSC 6809 EXTENDED BASIC may be made quite simple through the use of several expedients, listed below:

1. Code the function using position-independent techniques.
2. Transliterate the machine language code into DATA statements, coding the individual bytes as hex strings.
3. Using READ statements, transfer the code into a string variable in the program.
4. Using PTR, DPOKE and DPEEK, place the byte address of the string at MEMEND-2.
5. Access the machine-language routine through function calls of the form USR (X%).
6. Within the routine, locate the 2-byte argument (if any) at MEMEND-4 and the resulting 2-byte value (if any) at the same address.
7. If an address or other 2-byte value greater than 32767 is to be passed as an argument to a USR function, use DPOKE and PTR to place the value at the location of an integer variable without conversion.

An example embodying these techniques follows.

```
23150 REM 6809 USER FUNCTION FOR ACIA I/O
[illegible] DATA "FE","CC","2B","AE","5C","27","17"
[illegible] DATA "EE","94","10","AE","02","27","1D"
[illegible] DATA "A6","C0","F6","E0","04","57","57"
[illegible] DATA "24","F9","B7","E0","05","31","3F"
[illegible] DATA "20","EE","B6","E0","04","47","24","FA"
[illegible] DATA "B6","E0","05","84","7F","A7","5D","39"
[illegible] DATA ""
[illegible] REM
[illegible] REM INPUT TO USER FUNCTION IS AS FOLLOWS:
[illegible] REM  GET CHARACTER -- ZERO
[illegible] REM  PUT STRING    -- POINTER TO STRING DESCRIPTOR
[illegible] REM OUTPUT FROM USR FUNCTION IS AS FOLLOWS:
[illegible] REM  GET CHARACTER -- CHARACTER (LAST 7 BITS ONLY)
[illegible] REM  PUT STRING    -- POINTER TO STRING DESCRIPTOR
[illegible] REM PARAMETER IS LOCATED AT MEMEND-4
[illegible] REM
[illegible] REM * ACIA ROUTINE FOR 6809   AND FOR 6800
[illegible] REM MEMEND EQU  $CC2B        MEMEND EQU  $AC2B
[illegible] REM ACIA   EQU  $E004        ACIA   EQU  $9004
[illegible] REM START  LDU  MEMEND       START  LDX  MEMEND
[illegible] REM        LDX  -4,U                DEX
[illegible] REM        BEQ  GETCHR              DEX
[illegible] REM PUTCHR LDY  $00,X               DEX
[illegible] REM        LDX  $02,X               DEX
[illegible] REM PUTCH1 BEQ  EXIT                STX  HOLDX
[illegible] REM        LDAA ,Y+                 LDX  $00,X
[illegible] REM PUTCH2 LDAB ACIA                BEQ  GETCHR
[illegible] REM        ASRB              PUTCHR STX  HOLDX
[illegible] REM        ASRB                     LDX  $02,X
[illegible] REM        BCC  PUTCH2              STX  LENGTH
[illegible] REM        STAA ACIA+1              LDX  HOLDX
[illegible] REM        LEAY -1,Y                LDX  $00,X
[illegible] REM        BRA  PUTCH1              STX  HOLDX
[illegible] REM GETCHR LDAA ACIA         PUTCH1 LDX  LENGTH
[illegible] REM        ASRA                     BEQ  EXIT
[illegible] REM        BCC  GETCHR              DEX
[illegible] REM        LDAA ACIA+1              STX  LENGTH
[illegible] REM        ANDA #$7F                LDX  HOLDX
[illegible] REM        STAA -3,U                LDAA $00,X
[illegible] REM EXIT   RTS                      INX
[illegible] REM        END                      STX  HOLDX
[illegible] REM                          PUTCH2 LDAB ACIA
[illegible] REM                                 ASRB
[illegible] REM                                 ASRB
[illegible] REM                                 BCC  PUTCH2
[illegible] REM                                 STAA ACIA+1
[illegible] REM                                 BRA  PUTCH1
[illegible] REM                          GETCHR LDAA ACIA
[illegible] REM                                 ASRA
[illegible] REM                                 BCC  GETCHR
[illegible] REM                                 LDAA ACIA+1
[illegible] REM                                 ANDA #$7F
[illegible] REM                                 LDX  HOLDX
[illegible] REM                                 STAA $01,X
[illegible] REM                          EXIT   RTS
[illegible] REM                          HOLDX  FDB  0
[illegible] REM                          LENGTH FDB  0
[illegible] REM                                 END
[illegible] QU$="":RESTORE 28200:QD%=0:QS$="":DPOKE PTR(QD%),PTR(QS$)
28800 READ QS$:IF QS$<>"" THEN QU$=QU$+CHR$(HEX(QS$)):GOTO 28800
28810 DPOKE DPEEK(HEX("CC2B"))-2,DPEEK(PTR(QU$)):REM 6809 USER FCN
28820 REM DPOKE DPEEK(HEX("AC2B"))-2,DPEEK(PTR(QU$)):REM 6800 USER FCN
```

```
100    REM            COMMAND INPUT FOR TSC'S EXTENDED BASIC
110    REM
120    REM       This routine uses TSC's INSTR function to get a
130    REM    command. Then ON GOTO is used to transfer control to
140    REM    that command.
150    REM
160    REM                 Frank Hoss
170    REM         Frank Hoss Dental Laboratory
180    REM          Syracuse, New York 13210
190    REM               315-474-7856
200    REM
210    REM
220  C$ = "--FIRSECTHIFOUFIFEND"         !REM The command string with 2 leading
230    REM                                          blanks.
240  PRINT
250  PRINT  "SELECT ONE OF THESE:"
260  PRINT
270  PRINT  "      FIRST ONE"
280  PRINT  "      SECOND ONE"
290  PRINT  "      THIRD ONE"
300  PRINT  "      FOURTH ONE"
310  PRINT  "      FIFTH ONE"
320  PRINT
330  INPUT  "COMMAND" , B$
340    REM
350    REM  allow a command to be sent to FLEX if preceded by a '+'
360    REM  You can even RENUMBER the program thats running ! !
370    REM
380  IF LEFT$(B$,1) = "+" THEN EXEC, MID$(B$,2)    !  GOTO 240
390    REM
400    REM  Make sure there is at least 3 characters
410    REM
420  IF LEN(B$) < 3 GOTO 590
430    REM
440    REM  Chop off all but first 3 characters.
450    REM
460  A$ = LEFT$(B$,3)
470    REM
480    REM  Make I% point to the position in the string that the
490    REM  match occured. INSTR returns a 0 if no match.
500    REM  Divide by three and add 1 to give a number 1-7
510    REM  for the ON GOTO to work properly.
520    REM
530  I% = INSTR(1,C$,A$) / 3 + 1
540    REM
550    REM          ERR   (1)   (2)   (3)   (4)   (5)   END
560    REM
570  ON I% GOTO  590 , 610 , 620 , 630 , 640 , 650 , 660
580    REM
590  PRINT "''";B$;"''  is not a valid command"      !  GOTO 330
```

```
600   REM
610 PRINT ' This is the first one'          : GOTO 240
620 PRINT ' This is the second one'         : GOTO 240
630 PRINT ' This is the third one'          : GOTO 240
640 PRINT ' This is the fourth one'         : GOTO 240
650 PRINT ' This is the fifth one'          : GOTO 240
660 PRINT ' This is the END'
670 END
```

SOME COMMONLY ASKED QUESTIONS AND ANSWERS ABOUT THE NEW MICROWARE 6809 SOFTWARE

Q: What is required to bring OS-9 up on my computer?

A: Tape-based versions require at least 4K RAM. Disk versions require 8K RAM. In both cases, twice the minimum amount is a good working amount. Our standard packages are set up to plug into most SS-50 bus machines: RAM from address 0 up, ROM at the top of memory (at least 2K - the other 2K can be anywhere), and I/O ports at $E000. An ACIA-type interface is assumed in port 0. For tape systems, a MC standard tape interface using an ACIA serial interface at $E010 is assumed. Disk systems expect to see the controller at an address that depends on the type of board (You need to specify which brand and model when you order). In some cases we'll include information on how to change switches or jumpers to set your I/O addresses to a most optimum location for memory expansion. Our ROM sets are designed for plug-compatible operation on SWTPC and GIMIX CPU cards and mainframes.

Q: What if my system has non-standard addresses, I/O equipment, or I simply prefer some other configuration?

A: OS-9 is inherently a very adaptable operating system. Also, if you change OS-9 you should not have to change any other programs in the computer. There is a table in ROM called a "configuration table" that contains a number of start-up addresses, device names, program names, etc. It is possible to reconfigure any part of the configuration by installing an alternate configuration table in RAM addresses anywhere in memory, which would contain your own parameters. In some cases the same ROM could also contain compatible device drivers for whatever particular I/O device(s) you will be using. Though this is fairly simple to do, the particulars are beyond the scope of this paper but are described fully in the OS-9 Systems Manual which is supplied with each program.

Q: Is OS-9 or its files compatible with my present DOS such as FLEX, DOS-68, etc.?

A: No. When you install OS-9 it entirely replaces the DOS and monitor ROM you presently use. We do plan to have utility programs available in the future to read text files from your old DOS to OS-9 files.

Q: If I want to go to multi-user operation later on, will I have to buy all new software? Will my present hardware work too?

A: OS-9 Level Two (to be introduced this summer) is an upward-compatible superset of Lev 1 One. Most software you buy from us for Level One is guaranteed to work on Level Two. Also, if you follow our recommendations in the OS-9 Systems Manual when writing machine-language programs, they will also be upward compatible. As for the hardware side of things, it depends on how new your computer is and what brand it is. OS-9 Level Two requires a CPU with dynamic address translation (DAT) hardware such as SWTPC or GIMIX, a real time clock of some sort, a minimum of 32K RAM and a disk controller that supports interrupt-driven or DMA data transfer.

Q: I like some of the software you are offering, but I prefer to stick with my old DOS. Will you offer versions for other operating systems?

A: In most cases, no. Many present 6809 operating systems could not support this type software anyway.

CLASSIFIED ADVERTISING

SWTPC MP-16, 16K MEMORY BOARD, EXCELLENT CONDITION, PAID $450 SELL FOR $299. CALL 702-453-4775 AFTER 5:00 PM.

***

SWTPC 680D, 20K AC30-$1000. TI725 "SILENT 700"-$750. AJ SELECTRIC PRINTER & TRACTOR-$1000. CALL ROB AT 303-237-2518.

***

HELP

PURCHASED 32-1000 (HARD SECTOR) 1 BOX, NEEDED 34-1000, MAKE TRADE OR CASH OFFER. R.D.#3 LAWSON RD., JAMESTOWN, N.Y. 14701, RUSS PINSBURY 716-484-0859.

***

BEING AN AVID READER & SUBSCRIBER OF 68 MICRO JOURNAL, I HOPE YOU MIGHT HAVE A READER WHO CAN HELP ME. DOES ANYONE HAVE THE PATCHES NECESSARY TO RUN SWTP 4K BASIC VER.2.0 WITH A MP-S SERIAL INTERFACE BOARD LOCATED AT PORT#1. I WOULD LIKE TO PURCHASE SWTP 8K BASIC VER 2.3 ON CASSETTE, ALSO NEED INFORMATION ON EPROMING BOTH SWTP BASICS AT ADDRESS C000. PAUL RAMOS, 100 MIDDLE ST.,WOBURN, MA 01801 617-935-3758 EVES & WKENDS.
P.S. WOULD LIKE TO HEAR FROM READERS IN BOSTON MASS VICINTY INTERESTED IN FORMING A 6800/09 USERS INFORMATION EXCHANGE CLUB.

SMOKE SIGNAL BROADCASTING

Presents

3 Powerful New SS-50/SS-50C Boards

DCB-4
Disk Master
Double Density Controller Board and DOS68D Double Density DOS
$449.00

The new DCB-4 is a truly state-of-the-art development which allows up to 366K bytes to be stored on a single 5¼" disk and has these outstanding features:

- Up to four 5¼"and four 8" drives can be handled in the same system with a user definable logical unit table. (DOS68D will be compatible with future hard disk systems).
- Under software control, the user can select the following for any drive:
  - ☆ Single sided or double sided operation.
  - ☆ Single density or double density data.
  - ☆ 5¼" or 8".
  - ☆ Stepping Rate.
  - ☆ 40 track or 35 track density on double sided 5¼" drives.
  - ☆ User can select the system boot configuration.
- Occupies only 16 bytes of memory space (F760-F76F standard). User selectable to any 16 byte address space.
- Can read and write a single sector by itself. On-board buffer memory allows full interrupt capability in interrupt driven systems. Once data transfer has been initiated, no more processor time is required.
- Contains extended decoding circuitry for extended addressing per SS–50C bus which can be enabled by an option jumper.
- SSB provides a means for copying software written by older versions of DOS68 to be read by DOS68D. All new media formatted by DOS68D can be read by all older versions of DOS68. DOS68 is SSB's 6800 disk operating system.
- Track Ø of side Ø is recorded in single density per IBM standard.
- Phase-locked-loop assures highest data integrity attainable.

All of these features are available for immediate delivery on one standard 5½" x 9" 50 pin SS–50/SS–50C card for only $449.00. The price includes DOS68D version 5.1, MONITOR object code on diskette, and a manual with the source listing.

SCB-69
Super Computer Board
6809CPU Board
$299.00

The most versatile 6809 CPU Board on the market is now available from Smoke Signal Broadcasting and has the following features:

- Standard 2 MHz operation. (Shipping 1.5 MHz until 68B09 available)
- 20 bit address generation for up to 1 Mbyte of memory. Uses an improved address translation RAM which is compatible with present extended addressing schemes yet requires much less overhead when used in multi-user systems.
- All on-board devices can be switch selected to occupy any or all extended pages. Any on-board device may be disabled and its memory space is then available for external memory.
- Standard real–time clock (time–of–day, day–of–week, day–of–month) with battery back up capable of generating programmable interrupts.
- Up to 20K of EPROM can be installed on the CPU Board.
- Standard 1K of RAM on board.
- Includes improved 6809 Monitor (and source listing).
- Contains an FPLA for decoding EPROM address and optional devices. Switches are used to select 2K/4K EPROM and Fast/Slow I/O.
- Contains provision for optional 9511/9512 floating point processor.
- NMI line is user selectable to work with either SS–50 or SS–50C busses.

Price for the new SCB–69 is only $299.00 for an assembled, burned-in fully tested board.

M-32-X
32K
Memory Board
$539.00 $439.00

The first and only 32K Static Ram Board on standard size (5½" x 9") SS–50/SS–50C Bus Circuit Card is made by Smoke Signal.

- Switch selectable to any 4K boundary.
- Any 4K block may be switch enabled or disabled.
- Fully compatible with SS–50C extended addressing (allows memory decoding up to 1 Mbyte).
- Extended addressing capability may be switched off for compatibility with SS–50 systems.
- Gold Bus Connectors for high reliability.
- Guaranteed 2MHz operation (tested at 2.2 MHz).
- Low power consumption – 8 volts at 2.4 amps typical.

M–32–X 32K Memory Board is priced at $539.00.

M–24–X 24K Memory Board expandable to 32K, is $439.00.

And our M–16–X 16K board is back to the old price of $299.00.

SMOKE SIGNAL BROADCASTING®

31336 Via Colinas, Westlake Village, CA 91361, (213) 889-9340

SOFTWARE FOR THE HARDCORE

We know you hardcore bit hackers will recognize the computing power derived from combining the **FORTH** language with the 6809, today's most advanced 8 bit microprocessor.

And we know you'll understand this machine's 16 bit math, indirect addressing and two stacks are ideally suited for implementing **FORTH**.

But...should anyone need further convincing that **FORTH** provides a new dimension in power, speed and ease of operation, consider the following:

- It's a modern, modular, structured-programming high-level compiled language.
- It's a combined interpreter, compiler, and operating system.
- It permits assembler code level control of machine, runs near speed of assembler code, and uses less memory space than assembler code.
- It increases programmer productivity and reduces memory hardware requirements.
- It replaces subroutines by individual words and related groups of words called Vocabularies. These are quickly modified and tested by editing 1024-character text blocks, called screens, using built-in editor.

tFORTH is a basic system implemented for SS-50 buss 6809 systems with the TSC FLEX 9.0 disk operating system. It is available on 5¼" or 8" single density soft-sectored floppy disks. **$100.00**

tFORTH + consists of tFORTH plus a complement of the following FORTH source code vocabularies: full assembler, cursor controlled screen editor, case statements, extended data types, general I/O drivers. **$250.00**

firmFORTH is an applications package for use with tFORTH. It provides for recompilation of the tFORTH nucleus, deletion of superfluous code and production of fully rommable code. **$350.00**

Call or write today.

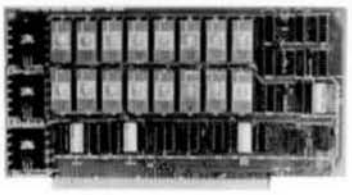
Also available for 6800

KENYON MICROSYSTEMS

10690 Shadow Wood, Suite 105 • Houston, Texas 77043 • (713) 464-0150

DIGITAL RESEARCH COMPUTERS
(214) 271-3538

32K S-100 EPROM CARD
NEW!



$74.95 KIT

USES 2716's

Blank PC Board - $34

ASSEMBLED & TESTED ADD $30

SPECIAL: 2716 EPROMs (450 NS) Are $19.95 EA. With A Ove Kit.

KIT FEATURES:
1. Uses +5V only 2716 (2Kx8) EPROM's
2. Allows up to 32K of software on line!
3. IEEE S-100 Compatible
4. Addressable as two independent 16K blocks
5. Cromemco extended or Northstar bank select
6. On board will state circuitry if needed
7. Any or all EPROM locations can be disabled
8. Double sided PC board, solder masked, silk-screened
9. Gold plated contact fingers
10. Unselected EPROM's automatically powered down for low power
11. Fully buffered and bypassed
12. Easy and Quick to assemble

16K DYNAMIC RAM PARTIALS

LOOK! INTEL 2108 8K X 1 RAMS LOOK!

8 FOR $9.95 32 FOR $35

FACTORY PRIME!

Huge special purchase of INTEL Dynamic RAM's. These are 2108-4, 300NS, 8K, Ceramic DIP. The 2108 is the INTEL 2116 (16K) tested for either upper or lower 8K only. These are factory prime. Full Spec. See INTEL 1978 Cat. for details or Memory Design Handbook for application data. Both IMSAI and EXTENSYS did mfg. S-100 RAM boards using these devices. — P.S. These devices will not work in the SD EPANDORAM™. Please specify upper or lower 8K. (S1626 or S1627). A super easy RAM to interface to a Z80, 16 PIN DIP.

FOR 4MHZ SALE! LOW POWER - 300NS 8 FOR

2114 RAM SALE! $44

4K STATIC RAM'S. MAJOR BRAND. NEW PARTS.

These are the most sought after 2114's. LOW POWER and 300NS FAST.

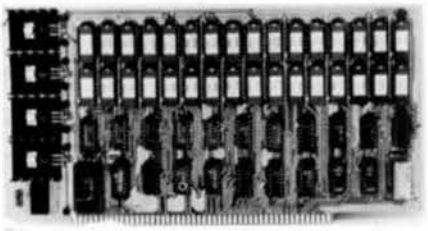
8 FOR $44

16K STATIC RAM KIT-S 100 BUSS

PRICE CUT!

$225 KIT

FOR 4MHZ ADD $10



KIT FEATURES:
1. Addressable as four seperate 4K Blocks
2. ON BOARD BANK SELECT circuitry (Cromemco Standard!). Allows up to 512K on line!
3. Uses 2114 (450NS) 4K Static Rams
4. ON BOARD SELECTABLE WAIT STATES.
5. Double sided PC Board, with solder mask and silk screened layout. Gold plated contact fingers
6. All address and data lines fully buffered
7. Kit includes ALL parts and sockets
8. PHANTOM is jumpered to PIN 67
9. LOW POWER: under 1.5 amps TYPICAL from the +8 Volt Buss
10. Blank PC Board can be populated as any multiple of 4K.

BLANK PC BOARD W/DATA-$33

LOW PROFILE SOCKET SET-$12

SUPPORT IC'S & CAPS-$19.95

ASSEMBLED & TESTED-ADD $35

OUR #1 SELLING RAM BOARD!

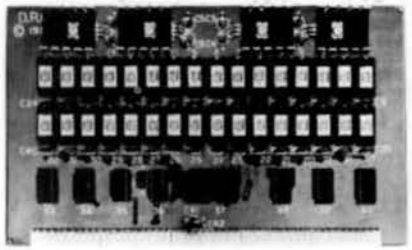
16K STATIC RAM SS-50 BUSS

PRICE CUT!

$229 KIT

FULLY STATIC!

FOR 2MHZ ADD $10



FOR SWTPC 6800 BUSS!

ASSEMBLED AND TESTED - $35

KIT FEATURES
1. Addressable on 16K Boundaries
2. Uses 2114 Static Ram
3. Fully Bypassed
4. Double sided PC Board. Solder mask and silk screened layout
5. All Parts and Sockets included
6. Low Power: Under 1.5 Amps Typical

BLANK PC BOARD—$30 COMPLETE SOCKET SET—$12

SUPPORT IC'S AND CAPS—$19.95

NEW! **STEREO! S-100 SOUND COMPUTER BOARD** NEW!

At last, an S-100 Board that unleashes the full power of two unbelievable General Instruments AY3-8910 NMOS computer sound IC's. Allows you under total computer control to generate an infinite number of special sound effects for games or any other program. Sounds can be called in BASIC, ASSEMBLY LANGUAGE, etc.

KIT FEATURES:
- TWO GI SOUND COMPUTER IC'S
- FOUR PARALLEL I/O PORTS ON BOARD
- USES ON BOARD AUDIO AMPS OR YOUR STEREO
- ON BOARD PROTO TYPING AREA
- ALL SOCKETS, PARTS AND HARDWARE ARE INCLUDED
- PC BOARD IS SOLDERMASKED, SILK SCREENED, WITH GOLD CONTACTS
- EASY, QUICK, AND FUN TO BUILD WITH FULL INSTRUCTIONS
- USES PROGRAMMED I/O FOR MAXIMUM SYSTEM FLEXIBILITY

Both Basic and Assembly Language Programming examples are included

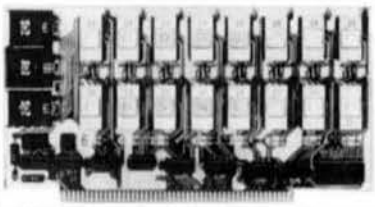
COMPLETE KIT! $84.95 (WITH DATA MANUAL)

BLANK PC BOARD W/DATA $31

SOFTWARE:

SCL™ is now available! Our Sound Command Language makes writing Sound Effects programs a SNAP! SCL™ also includes routines for Register Examine-Modify, Memory-Examine-Modify and Play-Memory. SCL™ is available on CP/M* compatible diskette at 2708 or 2716. Diskette - $20.00 2708 - $19.95 2716 - $29.95 Diskette includes the source. EPROM'S are ORG at E000H.

16K EPROM CARD-S 100 BUSS



$59.95 KIT

BLANK PC BOARD - $28

USES 2708's!

Thousands of personal and business systems around the world use this board with complete satisfaction. Puts 16K of software on line at **ALL TIMES!** Kit features a top quality soldermasked and silk-screened PC board and first run parts and sockets. Any number of EPROM locations may be disabled to avoid any memory conflicts. Fully buffered and has WAIT STATE capabilities.

ASSEMBLED AND FULLY TESTED — ADD $30

OUR 450 NS 2708'S ARE $8.95 EA. WITH PURCHASE OF KIT

RCA CMOS COMPUTER CHIP SET

INCLUDES:

1-CDP1802CD CPU	1-CDP1861CD VIDEO IC
2-CDP1822CE 256 x 4 RAM	1-CDP1862CE COLOR GEN.
1-CDP1858CE 4 BIT LATCH	1-CDP1863CE SOUND GEN.

COMPLETE SET $45 LIMITED QTY

NEW! G.I. COMPUTER SOUND CHIP

AY3-8910. As featured in July, 1979 BYTE! A fantastically powerful Sound & Music Generator. Perfect for use with any 8 Bit Microprocessor! Contains 3 Tone Channels, Noise Generator, 3 Channels of Amplitude Control, 16 bit Envelope Period Control, 2-8 Bit Parallel I/O, 3 D to A Converters, plus much more! All in one 40 Pin DIP. Super easy interface to the S-100 or other busses.

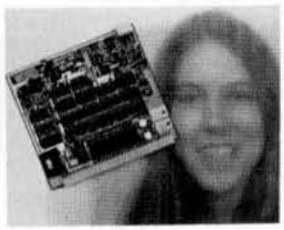
SPECIAL OFFER: $14.95 each Add $3 for 60 page Data Manual.

Digital Research Computers
(OF TEXAS)

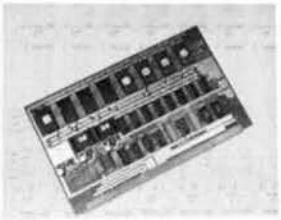
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Add $1.25 postage. We pay balance. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. 90 Day Money Back Guarantee on all items. Orders over $50, add 85¢ for insurance.

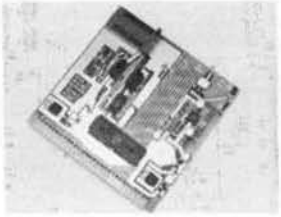
*TRADEMARK OF DIGITAL RESEARCH. NOT ASSOCIATED WITH DIGITAL RESEARCH OF CALIFORNIA, THE SUPPLIERS OF CPM SOFTWARE.



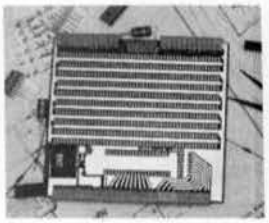
DS-68 DIGISECTOR



PSB-08 PROM SYSTEM BOARD



B-08 EPROM PROGRAMMER



UIO UNIVERSAL I/O BOARD

INNOVATIVE PRODUCTS FOR 6800 USERS

DS-68 DIGISECTOR is a random access video digitizer featuring 256 x 256 picture element scan and 64 levels of grey scale, with conversion times as low as 3 microseconds per pixel. It accepts either interlaced (NTSC) or non-interlaced (industrial) video input. Use it for computer portraiture, moving target indicators, precision security systems, fast to slow scan conversion. . . with clever software, the Digisector can read just about anything. Truly a professional tool at a price you can afford. $169.95

DM-85 DISK MIXER is an add-on board for the Smoke Signal Broadcasting BFD-68A Disk Controller which allows operation of both 8" and 5" drives. Controller mode (8" or 5") is selected on a drive-by-drive basis, so any mix of 5" and 8" drives is allowable. The 2" x 3" PC board mounts inconspicously on the back of the BFD-68A. Its operation is completely transparent to software. An oscilloscope is required for the setup procedure. Kit Price: $39.95

B-08 2708 EPROM PROGRAMMER is a compact unit that fits in the 6800's I/O slot. A safety switch and LED indicator provide control over the high programming voltage generated on board. An industrial quality Textool socket and extended board height allow effortless PROM insertion and retrieval. Fully commented source listings of U2708 is included in the Owner's Manual. $99.95

U2708 utility for testing, burning, verifying and copying 2708s in EPROM. $29.95

M6809 EMULATOR is a machine language program that will emulate all of the functions of the Motorola 6809 third generation microprocessor. Developed for use on any 6800 system, the program allows software development and debugging. The 3K byte program is complete with a 6809 mini-monitor and single-step trace routines. Fully commented source listing included. Specify Smoke Signal Broadcasting or FLEX™ disk, or KCS cassette. $49.95

PSB-08 PROM SYSTEM BOARD features 1K of high speed, low-power RAM and space for up to 8 2708 EPROMs, both DIP-switch addressable to start on any 8K boundary in memory. The exclusive I/O select feature allows you to move I/O locations up to any unused 1K block in the EPROM memory space. This permits memory expansion to a full 56K of contiguous user RAM. $119.95

UIO UNIVERSAL I/O BOARD helps you with your custom interfaces. It has space for a 40-pin wire wrap socket into which you may plug any of Motorola's 40 or 24-pin interface chips. All data and control lines are connected to the appropriate edge connector pins. All other bus connections are brought out to a 16-pin socket pad. + 5 volt regulator and all Molex connectors are provided; regulated + 5 and ground are bused among the locations for up to 35 14-pin ICs. $24.95

THE MICRO WORKS **P.O. BOX 1110, DEL MAR, CA 92014 714-942-2400**

HEMENWAY ASSOCIATES
SOFTWARE SOURCE BOOKS™

Your 6800 is up and running with HEMENWAY ASSOCIATES' complete software system. Software Source Books provide a powerful yet extensible programming package for business, scientific, or personal uaea.

Combining detailed descriptions with COMPLETE SOURCE CODE LISTINGS, these books explain the internal operations and algorithms used in HEMENWAY ASSOCIATES' popular systems software.

Imagine getting a complete 6800 software library, and at these suprisingly low prices.

Remember, these are not just books; they are Software Source Books™ complete software resources! Order them today; VISA and MASTERCHARGE accepted.

CP/68 OPERATING SYSTEM

The moat powerful operating system available for the 6800 family of microprocessors, this disk-baaed system features great flexibility. The user can add commands for special purposes. A single transient Peripheral Interchange Program (PIP) transfers data between devices. The system ia relocatable anywhere in memory and fits in leas than 8K. Other features include device-independent I/O and dynamic file allocation.

U.S. $34.95
Int. $52.45

XA6809
MACRO LINKING CROSS-ASSEMBLER

This new two-pass program generates relocatable and linkable code (requires LINK68). Resident on any 6800 system, XA6809 lets you produce code for a 6809 right now. This assembler has full macro facilities and features a COMMON section for the production of ROMable code. Conditional Assembly and fast execution.

U.S. $24.95
Int. $31.50

LINK68 LINKING LOADER

This ia a one-paaa linking loader which allows separately translated relocatable object modulea to be loaded and linked together to form a aingle executable load module, and to relocate modules in memory. It produces a load map and a load module in Motorola MIKBUG loader format. This book provides everything necessary for learning about this system and the nature of linking loader design in general.

U.S. $7.95
Int. $11.95

RA6800ML RELOCATABLE
MACRO ASSEMBLER

This two-paaa assembler produces a program listing, a sorted symbol table listing, and relocatable object code. The object code ia loaded and linked with other assembled modules using LINK68. This book fully describes the 6800 aasembly language and all major routines used, and includes flow charts, details on interfacing the assembler. Cross-referenced, showing all calling and called-by routines, pointers, flags and temporary variables.

U.S. $24.95
Int. $37.45

Structured BASIC language
STRUBAL+™COMPILER

The compiler features variable precision from 4 to 14 digits tor business or scientific uses and Structured Programming forms. It produces Relocatable and linkable code. You can create data structures with mixed data types, COMMON and DUMMY sections. STRUBAL+™ includes a complete scientific package. It allows tor string-handling and ia extensible.

U.S. $49.95
Int. $74.95

Please send the following items:
_____ copies CP/68 OPERATING SYSTEM
_____ copies STRUBAL+™ COMPILER
_____ copies XA6809 CROSS ASSEMBLER
_____ copies RA6800ML MACRO ASSEMBLER
_____ copies LINK68 LINKING LOADER

HEMENWAY ASSOC., INC.
101 TREMONT STREET
BOSTON, MA 02108
(617)426-1931

FOR NORTH AMERICA, ADD $.75/BOOK POSTAGE & HANDLING or $1.50/BOOK FOR FIRST CLASS
ALL OTHER DESTINATIONS, ADD $2.00/BOOK POSTAGE & HANDLING or $3.50/BOOK FOR PRIORITY MAIL

Name _____ Title _____ Company _____

Street _____ City _____ State/Country _____ Postal Code _____

☐ CHECK ENCLOSED IN THE AMOUNT OF $
☐ BILL VISA ☐ BILL MASTERCHARGE
CARD NUMBER _____ EXP. DATE _____
☐ FIRST CLASS/PRIORITY ☐ BOOK RATE

ATTENTION!

HOBBYISTS, EXECUTIVES, INVENTORS, ENGINEERS, PROGRAMMERS, SMALL BUSINESSMEN

LOOK AT WHAT IS AVAILABLE FROM STOCK
FROM THOMAS INSTRUMENTATION!

OUR NINTH YEAR IN BUSINESS

FEATURING

* * * * * NEW 16K (4-4K) MEMORY * * * * *

SS-50

*A "Cents"able way to add memory to your system
*Four 4K blocks individually addressable 0-F
*Additional memory at less than $10.00 per 1K
*Add memory 1K at a time, using low cost 2114's
*The Memory Card is available three ways

- Asm. & tested, socketed with all 16K **$295.00**
- Asm. & tested, socketed for 16K, with 1K **$129.00**
- Bare card and Documentation **$ 44.00**

SS-50

• • SPECIAL: $242.00 VALUE FOR ONLY $175.00 • •

- A set of our bare cards to build a small system
- Consists of 8-Slot Backplane/Motherboard,
- Super CPU, Video Ram, 16K Memory, 10 Port
- Parallel I/O, Wire Wrap Prototype Card, and
- Documentation for each of the above

• • OUR SS-50 LINE-UP • •

- All Thomas Instrumentation's assembled cards are burned in at 150°F and fully tested
- All cards come with full documentation including software source listings where appropriate
- Bare card price does not include edge connectors
  - Super CPU asm. with monitor source but without 2K-2708's EPROM monitor **$195.00**
  - Monitor in two 2708's EPROMS **$ 29.00**
  - CPU bare card, doc., & source **$ 49.00**
  - Video ram asm. 7X9 char. 64X16 line **$169.00**
  - Video ram bare card, doc., & source **$ 45.00**
  - Parallel I/O asm. 100 I/O lines includes 5 PIA's for 10 ports **$110.00**
  - Parallel I/O bare card & doc. **$ 35.00**
  - Wire-Wrap/Prototype bare card **$ 29.00**

• • NEW PRODUCT LINES • •

- NEW RCA sealed Keyboards
  - Model 611 **$ 85.00**
  - Model 601 **$ 65.00**
  - CPU Cable **$ 10.00**
- Leedex Monitor **$139.00**

• • NEW BACKPLANES/MOTHERBOARDS • •

- The following cards are extra thick (3/32)
  - 16 Position SS-50 **$80.00**
  - 12 Position SS-50 **$60.00**
  - 8 Position SS-50 **$40.00**
  - 4 Position SS-50 **$20.00**
  - 8 Position SS-30 **$39.00**
- SS-50 to SS-30 Transition card will be available next month
- Connectors for the above cards are separate, SS-50 take 5 for each pos., SS-30 take 3 each backplanes take males, main cards take females
- Males Tin $0.40ea. Gold $1.60ea.
- Females Tin $0.50ea. Gold $1.60ea.

DEALERS FOR SWTPC, GIMIX, AND TSC

WE DESIGN HARDWARE

WE WRITE SOFTWARE

THOMAS INSTRUMENTATION SPECIALIZES IN HELPING YOU DEVELOP LOW COST SYSTEMS TO MEET YOUR INDIVIDUAL COMPUTING NEEDS . . . LET US AUTOMATE YOUR LABORATORY
WE HAVE SPECIAL SYSTEMS AND PRICES FOR SCHOOLS

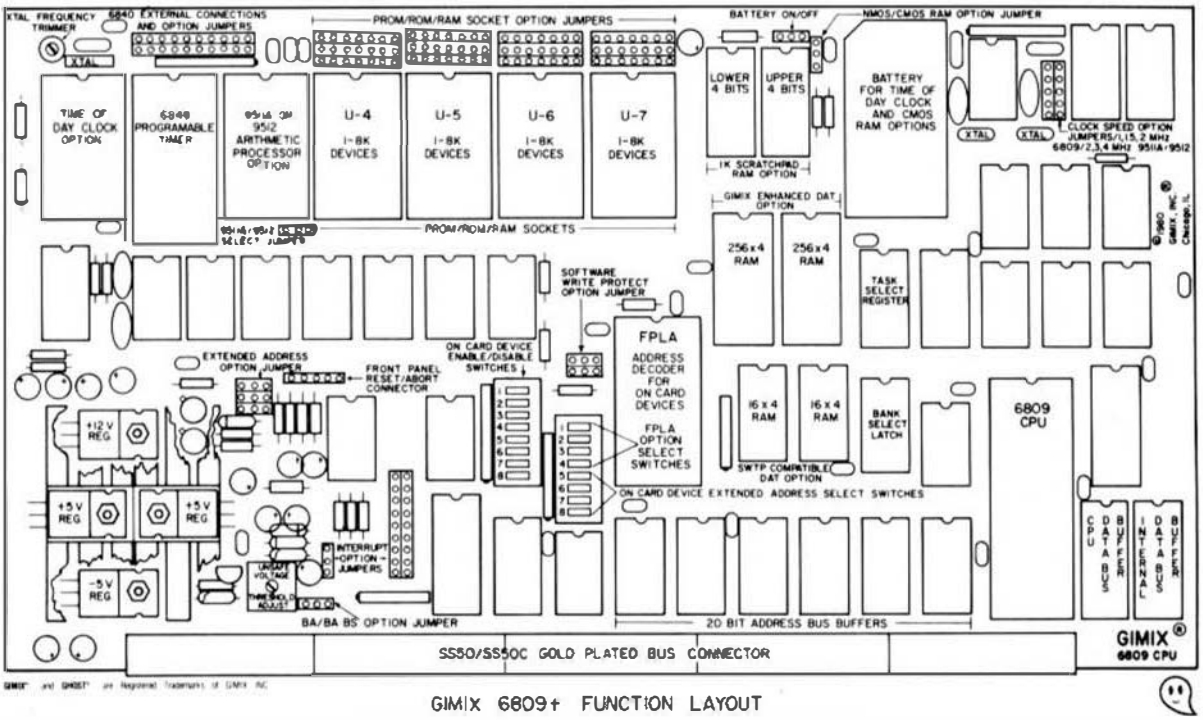
THOMAS INSTRUMENTATION

168 EIGHTH STREET AVALON, N.J. 08202 (609) 967-4280
N.J. RES. INCLUDE 5% SALES TAX
CONTINENTAL U.S.A. INCLUDE $2.00 SHIPPING, CANADA $5.00, FOREIGN $10.00

VISA

master charge

GIMIX PROUDLY PRESENTS!!!



GIMIX 6809+ FUNCTION LAYOUT

SS50C 6809 CPU

The 6809 CPU card will be available in a standard version and our **6809 PLUS** version that is fully socketed to allow adding options at anytime.

- + A 6840 timer package that provides 3 independent 16 bit counters is included on all **6809 PLUS** cards.
- + A 9511 or 9512 Arithmetic Processors option with its own independent crystal that allows you to use 2, 3, or 4 MHz parts in any combination with the 6809 running at 1, 1.5, or 2 MHZ.
- + 1K of scratchpad RAM
- + A Time of Day Clock option with battery back-up. With this option you can also substitute 1K of CMOS RAM that will also be battery backed up.
- + User selectable processor speeds without having to change the crystal.

32K of PROM, ROM or RAM. Both versions have 4 sockets that can each hold from 1K to 8K parts. Single or multiple voltage parts can be used on the **PLUS** version. The standard version only allows the use of single voltage parts.

All on board devices and options can use extended addressing so that they will only respond to that page to which they are set.

The card is double buffered and allows versatility in the use of software and memory address control disciplines.

Please note that this card does not have an on board baud rate generator, and must be used in systems where baud rates (if needed) are provided elsewhere in the system.

GIMIX INC. 1337 WEST 37th PLACE • CHICAGO, IL 60609 • (312) 927-5510 • TWX 910-221-4055

GIMIX® and GHOST™ are registered trademarks of GIMIX Inc. 1980 GIMIX Inc.

SS-50 MEETS UCSD PASCAL

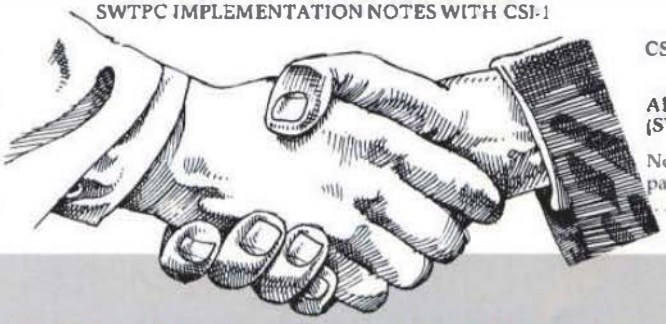
FOR 6809 SYSTEMS

IMMEDIATE DELIVERY FOR SWTPC, GIMIX USERS, 8" OR 5¼" DISKETTES
SMOKE SIGNAL BROADCASTING USERS, INQUIRE

FREE! UCSD PASCAL USERS MANUAL PLUS
SWTPC IMPLEMENTATION NOTES WITH CSI-1

CSI-1 O/S, PASCAL Compiler, Screen Editor, YALOE (Line editor for hard-copy terminals), Filer, Linker, Library, Setup, Binder, Interpreter, BIOS **$250.00**

CSI-2 BASIC Compiler, L-2 Editor, Patch, Disassembler, Calculator. **$100.00**



CSI-3 MARCO Assemblers for 6809 and 6800 ..... **$100.00**

ALL THREE DISKS AND MANUAL (SYSTEM) $419.00

Now Available! Pascal user-group compatible (IBM 3740) Disk Driver Routines **$20**

(SWTPC equipment compatible)

UCSD PASCAL™ MEETS BUSINESS

NEW MICRO-WINCHESTER DISK DRIVE FOR MICROCOMPUTERS

CSI announces a new option for their UDS 470 microcomputer: the micro-winchester hard disk drive. Available 4Q 1980 the micro-winchester drive will replace one of the two mini-floppy drives now standard with the UDS 470, thereby bringing a dramatic increase in on-line storage capacity.

This increased storage — 6.38 Megabytes — makes the UDS 470 viable for business applications requiring more on-line storage than previously available with floppy disk drives.

The micro-winchester drive will fit directly into the existing UDS 470 cabinet. Thus customers will realize the benefit of increased storage capacity without the disadvantage of finding space for a larger cabinet or an additional disk drive enclosure.

CALL TOLL-FREE (800) 255-4411

Continental U.S.A. only. (Kansas residents call (913) 371-6136)

CSI CONTROL SYSTEMS, INC.
MICROSYSTEMS DIVISION

master charge
VISA
for software only

1317 Central, Kansas City, KS 66102 (913) 371-6136 • 5200 West 73rd St., Minneapolis, MN 55435 (612) 831-0214
Drawer EE, Williamsburg, VA 23185 (804) 564-9350

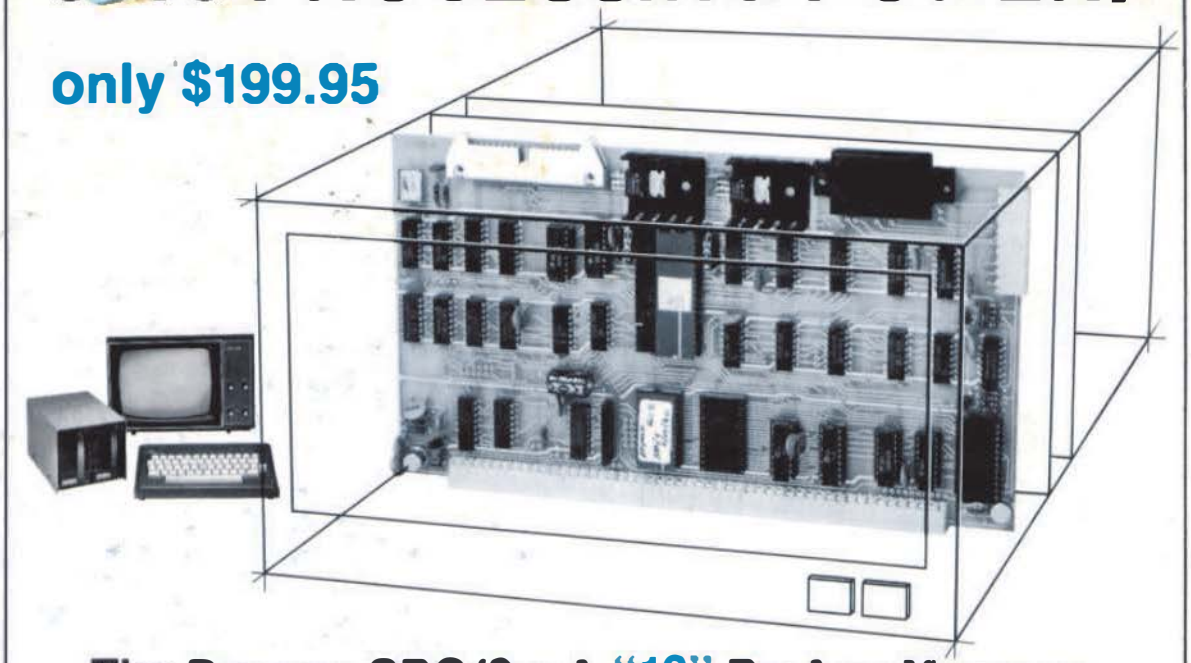
"UCSD Pascal" is a registered trademark of The Regents of the University of California.

'68' Micro Journal
3018 Hamill Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025

6809 PROCESSING POWER!

only $199.95



The Percom SBC/9™: A "10" By Any Measure.

Available with either the new, powerful 6809 μP or an optional 6800-software-compatible 6802, here are 10 beautiful reasons why the Percom SBC/9™ is not just another runner-up MPU/Single-Board-Computer card.

1. SS-50 bus direct, plug-in-compatible upgrade MPU. Requires no modification of the system bus, I/O or memory.
2. Full-capability stand-alone single-board computer. Accommodates a 6809 microprocessor or optional 6802 microprocessor **without modification.**
3. On-card 1 K ROM monitor "auto-links" to optional second 1 K PROM — if installed. Second PROM may be used to easily extend or modify the primary monitor command set.
4. Eight-bit parallel port is multi-address extension of system bus. Accommodates an exceptional variety of peripheral devices ranging from game paddles and keyboards to memory management modules. Connector is optional.
5. Serial port includes a full-range selectable bit rate generator. Optional subminiature 'D' connector provides RS-232 compatibility.
6. Extendable addressing via SS-50 bus baud lines to 1 Mbyte. Extendable addressing to 16 Mbytes or more through the parallel "super port."
7. Includes 1 Kbyte of static RAM.
8. All on-card I/O is fully decoded so that adjacent memory space may be used.
9. ROM circuit may be jumper-wired for single- or triple-voltage 2716 EPROM.
10. On-card power regulators simplify power supply design by minimizing regulation demands.

Plug the SBC/9™ into your SS-50 system bus, and just that easily you've upgraded to the new super-fast super-powerful 6809 MPU with such programming amenities as 10 addressing modes, 16-bit instructions, auto-increment/auto-decrement and position-independent code. Plus, you now have extended addressing capability, and operation under control of PSYMON™, the most powerful and flexible 1K ROM 6809 operating system yet written.

Percom SYstem MONitor PSYMON™ provides the usual ROM monitor functions in 1 Kbyte. It is easily extended and customized because its unique "look-ahead" program structure first searches an alternate command table. The table, if present, may be used to redefine or extend PSYMON's™ command set.

And with PSYMON™, I/O is easily directed to any peripheral device — even a disk system — through a Device Control Block table located in memory. This allows you to leave the details of I/O software to the separate I/O device drivers.

A PSYMON™ ROM is included free with the purchase of an SBC/9™. The Users Manual includes a source listing.

The 1 Kbyte ROM monitor for the SBC/9™ 6802 option includes a primary set of typical 6800-compatible monitor commands. As for PSYMON™, the commands are easily extended or modified.

Products are available at Percom dealers nationwide. Call toll-free, 1-800-527-1592, for the address of your nearest dealer, or to order direct. Prices and specifications subject to change without notice.

™ trademark of Percom Data Company, Inc.

PERCOM DATA COMPANY, INC.
211 N. KIRBY GARLAND, TEXAS 75042
©214/272-3921

PERCOM